# ABSTRACT


Title of dissertation: A GPU-ACCELERATED, HYBRID FVM-RANS METHODOLOGY FOR MODELING ROTORCRAFT BROWNOUT

Sebastian Thomas
Doctor of Philosophy, 2013

Dissertation directed by: Professor James D. Baeder
Department of Aerospace Engineering



A numerically efficient, hybrid Eulerian-Lagrangian methodology has been developed to help better understand the complicated two-phase flowfield encountered in rotorcraft brownout environments. The problem of brownout occurs when rotorcraft operate close to surfaces covered with loose particles such as sand, dust or snow. These particles can get entrained, in large quantities, into the rotor wake leading to a potentially hazardous degradation of the pilots visibility. It is believed that a computationally efficient model of this phenomena, validated against available experimental measurements, can be a used as a valuable tool to reveal the underlying physics of rotorcraft brownout. The present work involved the design, development and validation of a hybrid solver that combines the numerical efficiency of a free-vortex method with the relatively high-fidelity of a 3D, time-accurate, Reynolds-averaged, Navier-Stokes (RANS) solver. For dual-phase simulations, this hybrid method can be unidirectionally coupled with a sediment tracking algorithm to study cloud development.

To explore the use of GPUs for RANS simulations, a 3D, time-accurate, implicit, structured, compressible, viscous, turbulent, finite-volume RANS solver was designed and developed in CUDA-C. Validation and verification of the GPU-based solver was performed for both canonical and realistic bench-mark problems on a variety of GPU platforms. In these test-cases, a performance assessment of the GPU-RANS solver indicated that it was between one and two orders of magnitude faster than equivalent single CPU core computations ( as high as 50X for fine-grain computations on the latest platforms ). For simulations involving implicit methods, a multi-granular technique was used that sought to exploit the

intermediate coarse-grain parallelism inherent in families of line-parallel methods like Alternating Direction Implicit (ADI) schemes coupled with conservative variable parallelism.

The validated GPU-RANS solver was then coupled with GPU-based free-vortex and sediment tracking methods to model single and dual-phase, model-scale brownout environments. A qualitative and quantitative validation of the methodology was performed by comparing predictions with available measurements, including flow field measurements and observations of particle transport mechanisms that have been made with laboratory-scale rotor/jet configurations in ground effect. In particular, dual-phase simulations were able to resolve key transport phenomena in the dispersed phase such as creep, vortex trapping and sediment wave formation. Furthermore, these simulations were demonstrated to be computationally more efficient than equivalent computations on a cluster of traditional CPUs - a model-scale brownout simulation using the hybrid approach on a single GTX Titan now takes 1.25 hours per revolution compared to 6 hours per revolution on 32 Intel Xeon cores.

# A GPU-ACCELERATED, HYBRID FVM-RANS METHODOLOGY FOR MODELING ROTORCRAFT BROWNOUT

by

Sebastian Thomas

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Dr. James D. Baeder, Chair/Advisor
Dr. John G. Leishman
Dr. Christopher Cadou
Dr. Johan Larsson
Dr. Ramani Duraiswami, Dean's representative

*To Priya,*
*For filling my life with love, laughter and pumpkin pie*

# Acknowledgements

I would like to express my deep sense of gratitude to my advisor Dr. James D. Baeder for his constant support and encouragement during the course of my doctoral studies. His immense breadth of knowledge is made all the more impressive by the humility and curiosity that continues to accompany it. It has been an absolute honor to work under his guidance.

I would also like to thank my dissertation committee members for taking time out of their busy schedules to examine my thesis and provide feedback on my work. For the valuable scientific insights they have given me, via challenging coursework or discussions, I will always be indebted to them.

I would like to thank Dr. Shreyas Ananthan for his guidance during the initial stages of my graduate studies. His patience as a mentor and his work ethic as a researcher has made him an inspiring role-model.

I have had the good fortune of always being surrounded by a large number of excellent friends and colleagues, without whom this journey would certainly have been far less enjoyable. This peer group has been very supportive over the years, constantly challenging me to expand my horizons and evolve as a researcher and human being.

I would also like to extend heartfelt thanks to my family for their unwavering love and support.

Finally, it would hardly be an overstatement to say that this work would not have been completed without the love and support of my lovely wife, Priya. As friend, critic, comedienne, moral compass, crisis manager, devil's advocate and powerpoint expert, she continues to amaze me every single day with her superpowers.

*"While weather did not appear to play a factor, visibility may have been severely limited by nightfall and a significant amount of dust, referred to as a "brown out," created by the helicopters landing and taking off..."*

— Barbara Starr (CNN Pentagon Correspondent)
*"Midair Helicopter Collision Kills 7 Marines"*
February 24, 2012 (www.cnn.com)

*"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase."*

— Gordon Moore (Intel Co-founder)
*"Cramming More Components onto Integrated Circuits"*
April 19, 1965 (Electronics Magazine)

*"Going forward, the critical need is to build energy-efficient parallel computers, sometimes called throughput computers, in which many processing cores, each optimized for efficiency, not serial speed, work together on the solution of a problem. A fundamental advantage of parallel computers is that they efficiently turn more transistors into more performance. Doubling the number of processors causes many programs to go twice as fast. In contrast, doubling the number of transistors in a serial CPU results in a very modest increase in performance at a tremendous expense in energy. More importantly, parallel computers, such as graphics processing units, or GPUs, enable continued scaling of computing performance in todays energy-constrained environment. Every three years we can increase the number of transistors (and cores) by a factor of four. By running each core slightly slower, and hence more efficiently, we can more than triple performance at the same total power. This approach returns us to near historical scaling of computing performance."*

— Bill Dally (Chief Scientist, NVIDIA)
*"Life After Moore's Law"*
April 29, 2010 (Forbes Magazine)

# Contents

# List of Figures

# 1

# Introduction

## 1.1 Rotorcraft Brownout

The phenomenon of brownout usually occurs when a rotorcraft operates near surfaces covered with loose sediment particles. These particles (*'dispersed phase'*) can get entrained into the rotor wake (*'carrier phase'*), potentially resulting in a dense dust cloud that can completely engulf the vehicle. A photograph illustrating an example of a helicopter encountering brownout conditions during a landing in the desert is shown in Fig. 1.1. The developing dust cloud below the helicopter is clearly observed. A similar phenomenon can occur in snowy environments and is known as whiteout. Fig. 1.2 is a schematic from [1] that lends some insight into the complexity of the phenomenon by illustrating the various mechanisms at play during the development of the brownout cloud.

When enough particles are entrained in this manner, the dispersed phase has the potential to cause visual obscuration and motion cue anomalies, disorienting the pilot and adversely affecting his ability to operate the vehicle safely. In

**Figure 1.1:** A brownout cloud developing around a helicopter attempting a landing maneuver. Courtesy: Optical Air Data System LLC

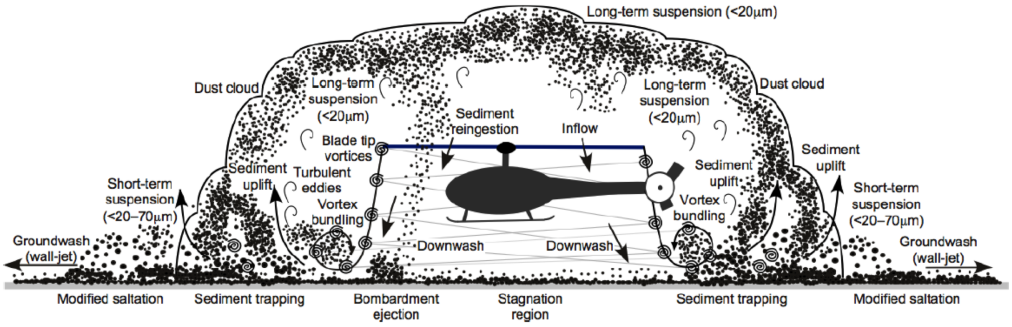



**Figure 1.2:** A schematic illustrating the different mechanisms involved in the development of a brownout cloud [1].

some cases, this may cause pilots to lose control of the helicopter, or drift into other obstacles, which can lead to mishaps. In fact, according to DOD estimates, encounters with brownout conditions are listed as the leading cause of human factor-related mishaps during military rotorcraft operations [2]. In addition, the formation of brownout clouds can result in issues such as rotor blade abrasion and

engine wear leading to significant decreases in component life-spans and increases in maintenance costs. Finally, the entrained dust particles can enter the vehicle and become a detrimental part of the internal cockpit environment.

Knowing the negative economical and ergonomical impacts associated with brownout, it is desired to have an improved understanding of the factors that influence the occurrence of the phenomenon. It is believed that this improved understanding can lead to mitigation strategies that ensure safe and cost-effective rotorcraft operations.

One technique currently being pursued is the use of sensor technology to allow for visual penetration of the surrounding dust clouds by projecting flight parameter values or sensory cues onto displays for easy access by the pilot during landing or take-off maneuvers [3, 4] . These systems have shown reasonable success at improving safety of flight. However, the installation of this sensor technology in the rotorcraft has the obvious effect of increasing operational and maintenance cost. Also, it does not mitigate the deleterious effects that the brownout cloud can have on the rotorcraft engine and crew.

Another operational tactic employed by pilots is the use of specialized landing and takeoff maneuvers to minimize the adverse effects of the developing brownout cloud [5]. An example of such a maneuver would be the minimizing of the time in flare during landing to prevent the entrainment of large quantities of dust into the rotor wake before touchdown. Such strategies are useful only to a limited extent since the precise behavior of the dust clouds that develop is an unknown

function of a large number of variables such as vehicle weight, particle characteristics, altitude, etc. that are almost never the same from one maneuver to the next. Furthermore, some of these strategies might involve an increase in risk to the pilot or the vehicle due to the resulting repetitive hard landings.

While both sensor technologies and specialized landing/takeoff maneuvers offer promise towards an increase in safety of flight, it is still desirable to find a more permanent solution to the problem of brownout. Since the interaction of the rotor wake with the ground and the dust particles form the root cause of the brownout problem, it is believed that a detailed understanding of the underlying fluid physics coupled with a knowledge of the fluid-particle interaction under these conditions can help in the development of effective means of preventing and/or mitigating the adverse effects of rotorcraft brownout. In Ref. 6, Milluzzo presents photographic evidence and preliminary technical calculations that suggest that different rotorcraft designs can produce significantly different brownout clouds. For example, some landing helicopters, in certain conditions, appear to produce toroidal brownout clouds of large radii, producing zones of good visibility for the pilot to pick up visual cues from. Figure 1.3(a) shows one example of a relatively benign dust cloud beneath a hovering EH-101 helicopter. Other rotorcraft are seen to exhibit large, dome-shaped brownout signatures that engulf the entire vehicle causing serious visibility degradation (Fig. 1.3(b)). Even nominally identical rotorcraft performing similar maneuvers can result in different brownout conditions depending on the characteristics of the loose sediment particles on the ground. Such observations suggest that the understanding of the underlying aspects involved in the formation of brownout clouds is a critical first

(a) Formation of a relatively mild brownout cloud beneath a hovering EH-101. Courtesy: AWI

(b) Severe brownout conditions beneath a hovering CH-47 [7]

**Figure 1.3:** Brownout signatures produced by two rotorcraft in different operating conditions

step towards attempting to mitigate the problem.

## 1.2 Previous Work - Analysis of Rotorcraft Brownout

### 1.2.1 Experimental Studies

The analysis of rotorcraft brownout over the last decade has involved both experimental and computational studies. On the experimental side, significant progress has been made in the analysis of two phase flow in controlled rotary environments.

For the experimentalists, the primary difficulty lies in simulating brownout conditions around a full-scale rotor. Despite this difficulty, there have been a few studies performed on full-scale configurations to characterize brownout clouds. One such study was DARPA's "Sandblaster" program [4] which consisted of field tests performed at the Yuma Proving Grounds. These tests extracted particle concentration and particle size distributions around helicopters performing hover-taxi maneuvers at various heights above the ground. The results from this study showed that the brownout clouds formed primarily contain small-sized particles with particles of larger diameters falling back onto the ground surface. In addition, this field test was able to extract a correlation between higher disk loading and higher cloud densities. Furthermore, the study showed that the concentration of the larger diameter particles was a function of the size of the rotorcraft airframe. Wong et al [8] used a photogrammetry technique to extract the size, convective velocity and other features from full-scale brownout clouds. The measurements from this study have been used to provide validation for some

computational techniques performed by other researchers [1]. One shortcoming of this data, however, is the lack of important information such as helicopter weight and blade control angles.

While the afore-mentioned studies have helped provide much needed insight into the phenomenon of full-scale rotorcraft brownout, it is difficult to extract details of the mechanisms involved in the actual entrainment of the particles at this scale. To overcome the problem arising due to scale and to have control over the environment, several studies have focused on studying single and dual-phase flowfields around laboratory-sized rotors.

Taylor et al [9] used flow visualization techniques to qualitatively measure the flow patterns produced by single and coaxial rotor systems operating in ground effect. Curtiss et al. [10] conducted an experimental and analytical study to investigate the aerodynamic characteristics of an isolated rotor IGE at low advance ratios. These experiments showed that the favorable effect on performance of ground proximity in hover disappears rapidly with small increases in advance ratio. Two flow regimes were shown to occur - a recirculation of the rotor wake at the low end of the advance ratio range and the formation of a ground vortex as the advance ratio was increased. Experimental results from this study also showed that translational acceleration has a significant effect on the ground effect of a lifting rotor. Hot wire measurements of the flow field under the rotor in ground effect were analyzed to determine effective values of the inflow in ground effect as well as the strength of the ground vortex, which was found to be stronger than the tip vortices by an order of magnitude.

Later, Light [11] used a wide-field shadowgraph method to perform flow visualization studies of rotor tip vortices produced by a hovering rotor operating in ground effect. This technique was used to extract detailed tip-vortex trajectories from both model-scale helicopter main rotors and tilt rotors, as well as full-scale tail rotors, both in hover and in forward flight. Nathan et al used 2D and stereoscopic PIV [12] to study the single- and dual- phase flow environments around a laboratory-scale rotor operating at different heights above ground and at different advance ratios. Analysis of the single-phase flow showed the formation of a recirculation region at low advance ratios, and the creation of a ground vortex at higher forward speeds, providing qualitative validation for the results by Curtiss et al [10]. This study also performed dual-phase flow visualization using fine talcum powder as the dispersed phase. It was observed that the talcum powder particles tended to accumulate near regions of high vorticity, such as the ground vortex or the recirculation zone.

Lee et al. [13] used high-speed flow visualization and particle image velocimetry (PIV) techniques to study the flowfield around a hovering laboratory-scale rotor positioned at various heights above a ground plane. The experimental setup used to study the single-phase flow is shown in Fig. 1.5(a). Time-averaged and phase-averaged velocity profiles close to the ground were extracted as part of this study. The significance of mechanisms such as vortex diffusion, vortex stretching, and turbulence generation were studied. Vortex stretching was shown to have the effect of increasing swirl velocities within the vortex cores leading to a reintensification of vorticity. It was found that these stretching effects were

largest when the rotor hovered at an intermediate height above the ground. Beyond this height, the effects of vortex diffusion became more significant than vortex stretching.

In 2009, Johnson [14] used an experimental setup similar to that of Lee and performed both single and dual-phase simulations of the flow environment induced by a small-scale rotor hovering above a sediment bed. High-resolution, near-wall, PIV measurements showed large excursions in the boundary layer velocity profiles due to the presence of the convecting vortices. The highest sediment entrainment levels were seen to occur at regions where the vortices were closest to the ground since this resulted in increased groundwash and wall shear in those regions. In addition, this study observed the highly aperioidic phenomena of vortex pairing and merging of adjacent turns of the blade tip vortices. These phenomena were found to play a key role in increasing local induced velocities, and consequently shear stresses, at the ground plane, which in turn increased initial mobilization and uplift of particles from the bed. The presence of a separation bubble just downstream of the vortex impingement zone was also observed. In the dispersed phase, the mechanisms of saltation, saltation bombardment, reingestion bombardment and vortex induced trapping were observed using high-speed flow visualization.

Milluzzo et al. [15] conducted rotor wake measurements, in ground effect, for rotors with various blade planforms and tip shapes. Four different blades were tested: rectangular, swept, BERP-like, and slotted-tip. These different planforms are shown in Fig. 1.4. One objective of this study was to study the flow patterns produced by the different blade shapes near the ground plane where particle en-

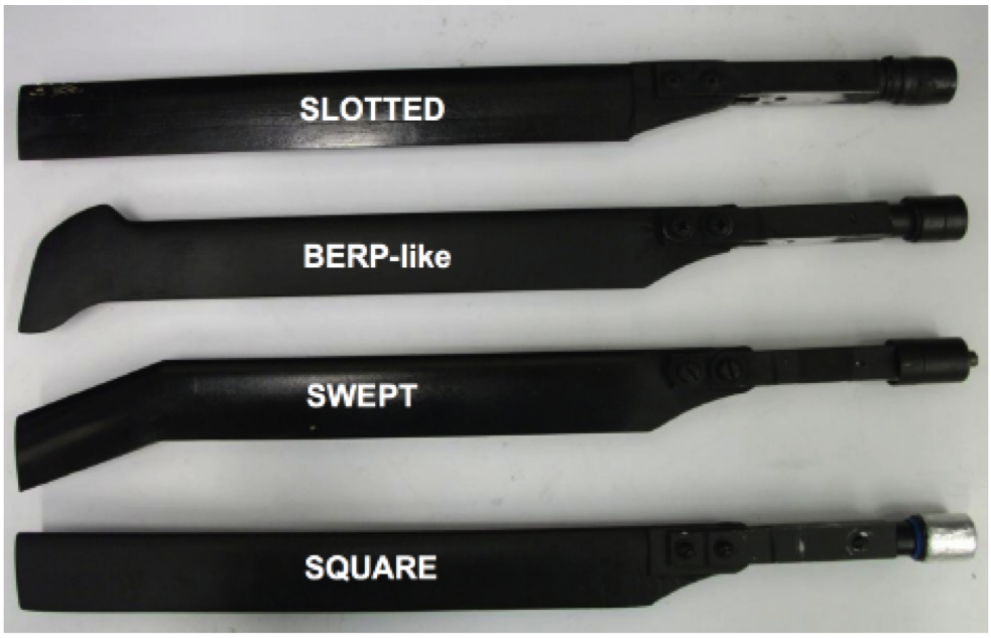
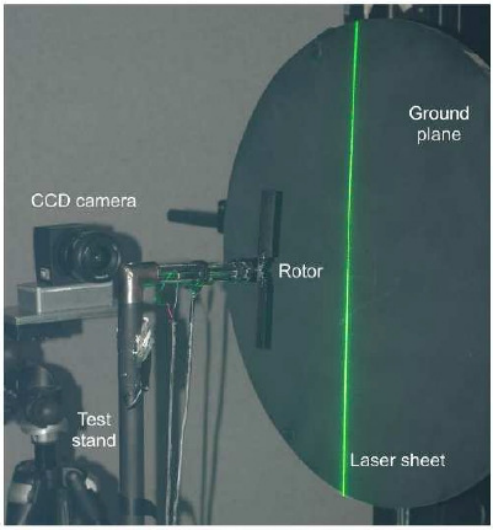



**Figure 1.4:** Different planforms/tip-shapes tested in the study by Milluzzo et al [15]

trainment is expected to occur. In addition, the core size and swirl characteristics of the tip vortices for the different blade planforms/tip-shapes were measured. One significant finding from this study was the effectiveness of the slotted-tip blade in reducing the strength and cohesiveneess of the tip-vortices through the acceleration of diffusive processes. This reduction in strength had the effect of reducing groundwash velocities (mean and excursions) at later wake ages which could play a key role in minimizing particle entrainment.

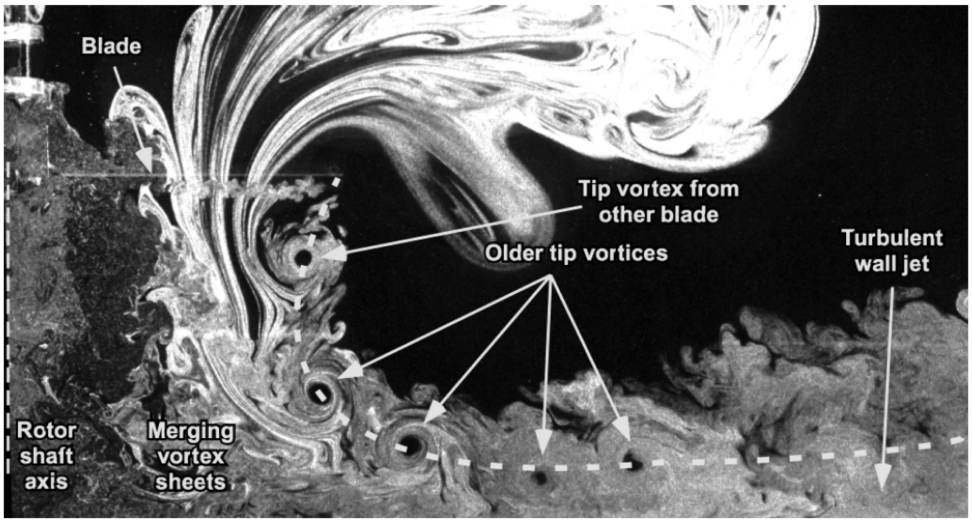
Sydney et al. [16] extended the analysis of Johnson by performing PIV and particle tracking velocimetry (PTV) measurements on the flow fields produced by 1- and 2-bladed rotors operating at identical blade loading coefficients in ground effect. The experimental setup used to study the dual-phase flow is shown in Fig.

1.5(b).




(a) Experimental setup used by Lee for single-phase flow studies [13]

(b) Brownout chamber used by Sydney et al for two-phase flow studies [16]




(c) Flow visualization imagine of a 2-bladed rotor operating in ground effect [16]

**Figure 1.5:** Experimental setup used to study single-phase/dual-phase flow beneath hovering rotors

The study demonstrated that the 2-bladed rotor exhibited enhanced aperiodicity due to the greater likelihood of vortex merging and pairing. In this study, several fundamental processes of sediment uplift and mobilization were identified:

1. Creep - particles slowly move along the sediment bed
2. Modified saltation bombardment - uplifted particles fall back to the ground and upon impact release additional particles from the sediment bed
3. Vortex trapping - uplifted particles are entrained into, and circle, individual vortices
4. Unsteady pressure effects - particles are uplifted in a direction perpendicular to the direction of jet flow due to a pressure gradient between the sediment bed and the vortex core
5. Reingestion bombardment - uplifted particles are accelerated around a vortex and then fall back to the ground causing the release of more particles from the sediment bed
6. Secondary suspension - particles are kept in suspension by being transferred from one vortex to the next

Mulinti et al [17] studied sediment dynamics in the presence of impinging jet flows and vortex rings - a simplified model for flow environments beneath hovering rotors (see Fig. 1.6). These studies used phase-locked flow visualization techniques and PIV to characterize these flows and to track the motion of the sediment particles. One key observation from this study was the production of secondary vortex structures due to the interaction of the vortex rings with the

ground plane.

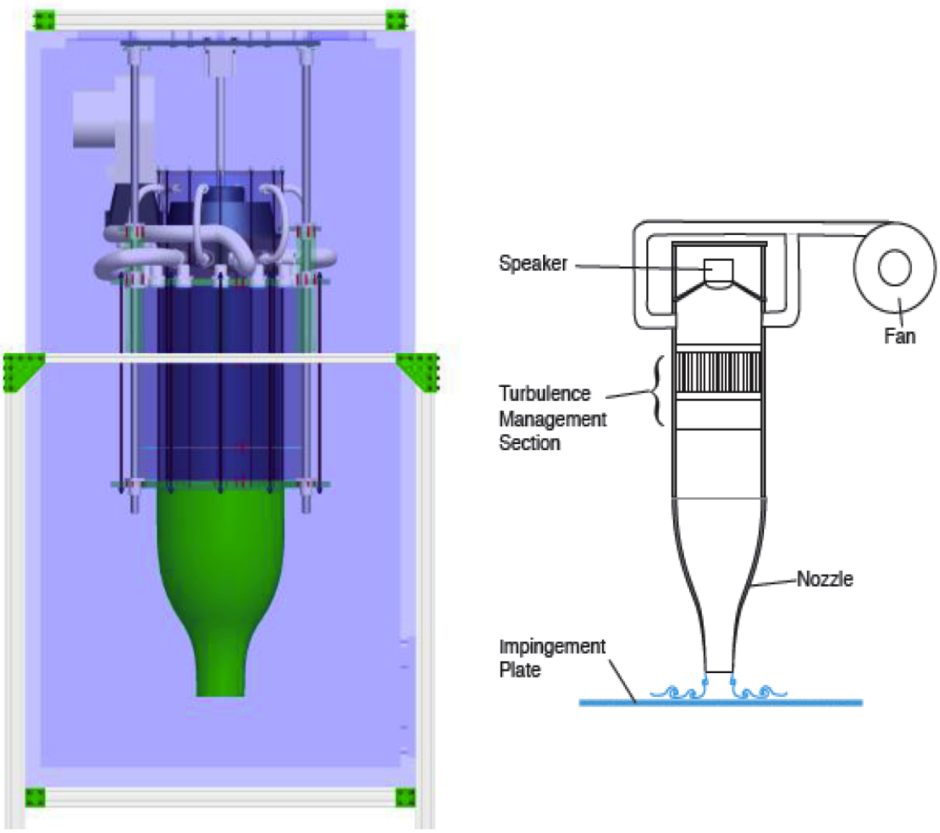



**Figure 1.6:** The experimental set-up inside the dust chamber used by Mulinti et al (left) and a labeled schematic of the set-up (right) [17]

Dade et al [18] performed experiments to study the response of a sediment bed beneath a hovering model-scale rotor, to the non-uniform spatial structure of the near-bed mean flow and Reynolds stresses. After the bed was exposed to the flow, changes in bed topography were analyzed to extract spatial patterns of sediment entrainment and transport.

### 1.2.2 Computational Studies

While experimental analysis is critical to understanding some of the key features of the underlying physics of the brownout problem, it can be very difficult and time consuming to conduct exhaustive parametric studies in the laboratory. In comparison to experiments, computational analysis can be used, with relative ease, to simulate a wide range of brownout conditions.

Brownout is an example of dual-phase flow, where the carrier fluid and the entrained particles interact with each other leading to interphase momentum and energy transfer. Any attempt to model this phenomenon, therefore, requires the coupling of a fluid dynamics solver with a sediment tracking algorithm. Most of the present research concerning brownout assumes a one-way interaction between the two phases; the particles are influenced by the flowfield but the fluid is unaffected by the particles. This is a reasonable approximation for dilute flows, where the mass fraction of the dispersed phase is not large, and the authors make use of this assumption in the present work. It must be noted that in reality, this approximation is inapplicable very close to the sediment bed.

Over the last few years, researchers modeling brownout have employed aerodynamic models of various levels of sophistication. Syal et al. [1, 19, 20] and Wachspress et al. [21] performed computational analysis of full-scale rotorcraft brownout clouds using a sediment tracking algorithm coupled with a time-accurate, free-vortex method (FVM) with an image plane to simulate the ground. The FVM is a very efficient numerical scheme for the preservation of far-wake

vortical structures. It requires an empirical model of the initial core size as well as how the vortex core diffuses with age as a function of vortex Reynolds number. The primary disadvantage of the free-wake method is that the image plane method used within it leads to the prediction of finite slip velocities at the ground plane, necessitating the use of empirical models to approximate the ground boundary layer. Figure 1.7 shows a representative free-vortex wake solution obtained using the image plane method.

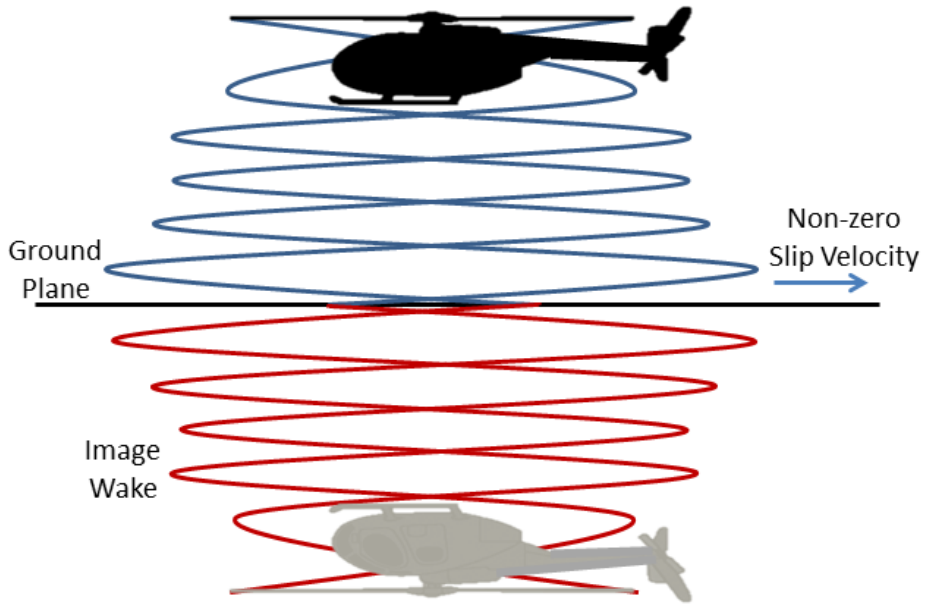



**Figure 1.7:** Representative free-vortex wake solution for a rotor operating in ground effect obtained using the image plane method

In addition, the Biot-Savart calculations in FVM, without a fast summation algorithm, exhibit a complexity of $O(n^2)$, implying that the run-time increases to

large levels when the wake discretization (or the number of field points at which induced velocities are needed to be computed) is increased. This problem has been largely circumvented through the use of parallelization and fast multipole algorithms [1].

In Ref. 22, an inviscid vorticity transport model (VTM) was coupled with a particulate transport model to generate brownout clouds around a generic, baseline, five-bladed rotor at different operating conditions. One advantage that the VTM has over Lagrangian methods like the FVM is that the small scale structures, which result from the breakdown of the larger vortical structures are properly resolved. Despite this, the methodology requires the approximation of the ground boundary layer. This is because, like in the FVM methods, the ground surface is modeled through the method of images leading to unrealistic, finite slip velocities at the ground plane. In addition to simulating the generic five-bladed rotor, Phillips et al also studied the effect of advance ratio, blade twist and number of blades on the size and severity of the developing brownout cloud. One key finding from this study was that the behaviour of the dust cloud, when the strength of the tip vortices was increased, was dependent on the advance ratio. At higher advance ratios, stronger tip vortices resulted in the dust cloud becoming larger and more dense.

Wenren et al [23] used a vorticity confinement method inside an incompressible flow solver to better resolve the tip vortices generated by a hovering UH-60. This study showed that the tip vortices tend to roll along the ground plane, creating concentric rings of increased particulate concentration. The use of the vorticity

confinement method allows for detailed flow structures to be resolved using relatively coarse meshes. However, this technique requires the use of empirical factors that may not be easily determined in ground plane simulations. Also, the vorticity confinement approach did not accurately resolve the ground boundary layer.

In 2010, Kalra et al. [24] used a high-fidelity overset, compressible, Reynolds Averaged Navier Stokes (RANS) solver to simulate the hovering micro-rotor setup of Lee et al., described earlier. The primary objective of this study was to demonstrate the capability of an overset-RANS methodology to provide good flowfield predictions for a hovering rotor operating close to the ground. The results were validated by comparison of the predicted integrated thrust and power coefficients with experimental data. The tip-vortex trajectory and time-averaged jet-flow predicted by the RANS simulations were also found to be in good correlation with experimental measurements. The primary disadvantage with RANS models is the computational expense associated with it - to accurately capture complex viscous phenomena such as boundary layer growth and vortex ground interactions a very large number of mesh points are required. In addition, for problems with vortical flows, a large number of mesh points are required to minimize artificial dissipation and allow for long term preservation of vorticity in the flow. Furthermore, the number of mesh points required for viscous, turbulent RANS simulations is known to scale superlinearly with Reynolds number, making full-scale, realistic simulations exceedingly expensive. Figure 1.8 shows the mesh system used in the study.

A total of 20.2 million points were used in this simulation. One consequence

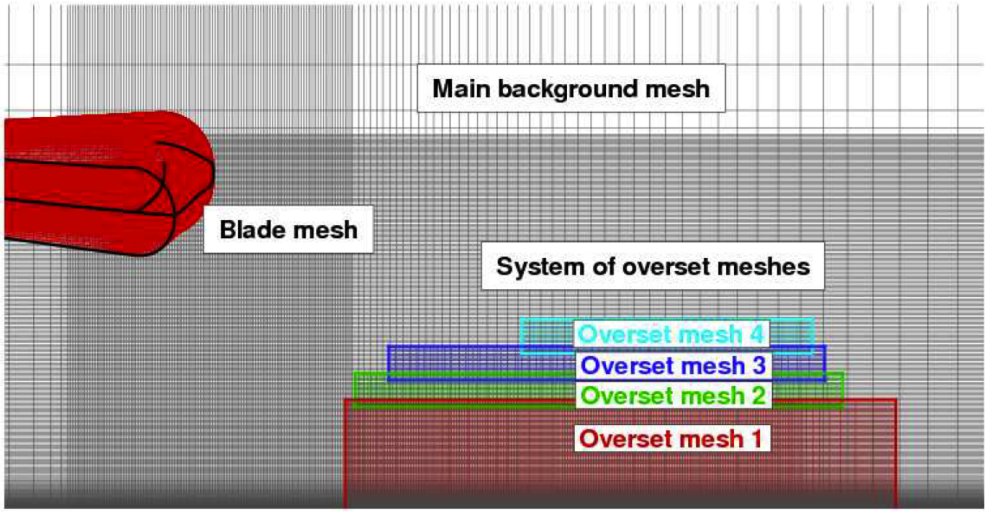



**Figure 1.8:** Overset mesh system used in the full RANS simulation by Kalra et al

of the large problem sizes in high-fidelity RANS simulations is that, for feasible runtimes, a parallelization strategy becomes necessary. The most popular strategy is what is referred to as the domain decomposition paradigm wherein different blocks of the mesh reside on different processors with information transferred between them at every iteration. Care is typically taken to ensure that the computational load is well-balanced to ensure that no processor is left idle. For peak performance, this method still needed to run on a large cluster of processors. One disadvantage with the domain decomposition paradigm is that, for a given mesh size, it is not scalable to an indefinitely large number of processors. The reason for this is that beyond a certain level of division, the time required for transferring data between processors becomes comparable to the time required for performing computations on each processor. In the present work, an attempt is made to circumvent this issue by pursuing a different parallelization strategy with GPU technology. The details of this approach are listed in Chapter 2.

Though the RANS methodology in [24] is relatively very expensive, unlike in the case of FVM or VTM, no boundary layer approximation is needed in this case. The aerodynamic model captures the boundary layer without the need for empiricism. This is very important for separated flow where the boundary layer profile is quite complicated. Figure 1.9 shows the flow environment near the ground plane beneath a hovering model-scale rotor. The separation of the ground

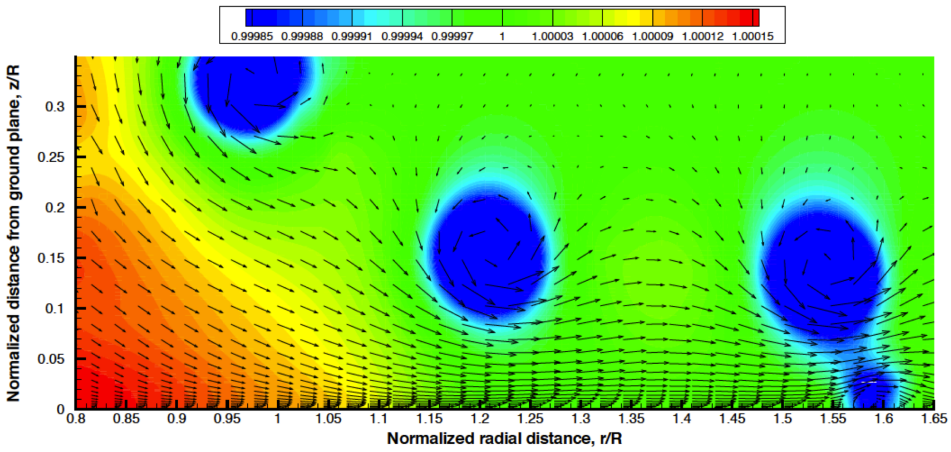



**Figure 1.9:** Velocity vectors along with pressure (non-dimensionalized by freestream value) contours for a rotor height of $h/R = 0.5$ [24]

boundary layer into a secondary vortex is clearly captured in this image. However this work focused on single-phase flowfield validation and a full brownout simulation was not performed as part of this study.

Morales [25] used a Direct Numerical Simulation (DNS) to gain insight into the fluid-particle dynamics near a sediment bed under the influence of coherent vortical structures. The competing effects of vortex interactions, inter-particle collisions and gravitational effects on particle transport were studied in detail.

A key finding in this study was the increased effects of inter-particle collisions near the boundary layer due to larger dispersed-phase concentrations in these locations caused gravitational settling.

Thomas et al. [26] extended the analysis of Kalra et al. to perform a one-way coupled, dual-phase brownout simulation by coupling the particle solver with a rotating, static CFD solution. The objective of this study was to demonstrate the capability of using a RANS solver with a reliable sediment code for particle transport to predict various transport mechanisms observed in the experiments of Sydney et al. [16]. The mechanisms of creep, saltation bombardment and vortex trapping were clearly captured in numerical experiments with various particle sizes. The disadvantage with this modeling approach is the inherent periodicity in the carrier phase due to the use of a static RANS solution. Experimental evidence suggests that, in reality, the flowfield beneath a hovering rotor is highly aperiodic.

In the present work, an attempt is made to combine the high computational efficiency of the free-wake method with the high-fidelity of a RANS solver. This hybrid Eulerian-Lagrangian model should be able to overcome the primary disadvantage of the free-vortex method in ground effect i.e. the prediction of unrealistic velocity profiles at the ground plane, without incurring the very large computational expense (due to a large number of mesh points) associated with RANS. Furthermore, the proposed method should be able to capture viscous phenomena near the ground such as boundary layer growth and separation, wall-jet formation and vortex-ground interaction. The details of this hybrid approach are presented in Chapter 2.

## 1.3 GPU computing

For several decades most computer simulations, such as the numerical solution of PDEs, have been executed in serial on single CPUs, or in parallel on a network of processors (for example, using Message Passing Interface (MPI) [27]), or on multicore CPUs in a single machine (for example, using OpenMP [28]). During this time, another computational platform that has evolved in parallel with the CPU is the graphics processing unit (GPU). The modern-day GPU is a specialized electronic circuit designed to accelerate the creation of images in a frame buffer intended for output to a display. The demand for real-time, high-definition 3D graphics, primarily from the gaming industry, has morphed the GPU into a highly parallel, multi-threaded, many-core processor with large computational power and memory bandwidth. Figure 1.10(a) shows the GeForce256, marketed by NVIDIA as the world's first GPU. It came with a clock speed of 120 MHz, a total VRAM of 32MB and a memory bandwidth of 2.6 Gb/s. The latest generation of GPUs is best represented by the GTX Titan, shown in Fig. 1.10(b), which was released in 2013 . It boasts a clock speed of 3100 MHz, a total VRAM of 6144MB and a memory bandwidth of 288 Gb/s. Over the fourteen years that separated the GeForce256 from the GTX Titan, this constitutes a 25X increase in clock speed, a 190X increase in VRAM and a 110X increase in bandwidth. In addition, the GeForce256 had only 4 cores per card and could handle only single precision floating point operations. The GTX Titan, on the other hand has 2688 cores and is capable of performing double precision computations.

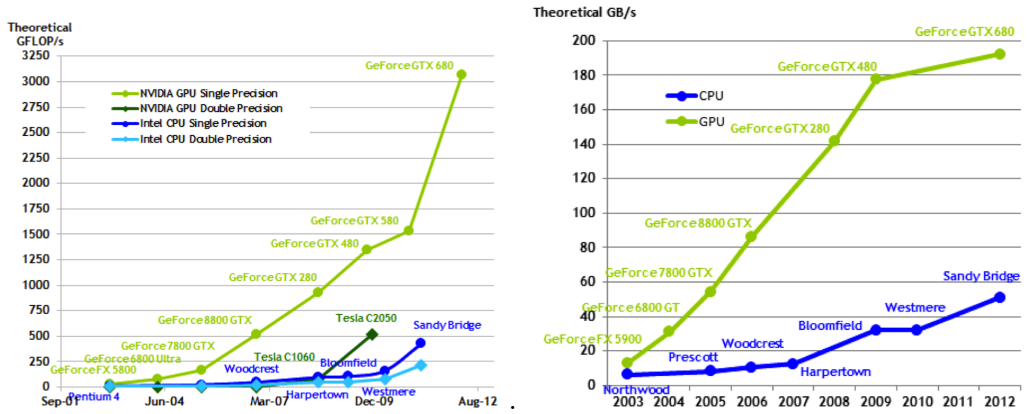
While GPU performance has increased dramatically in recent years as shown


(a) The NVIDIA GeForce 256, released in 1999, which was marketed as the world's first GPU with compute power of 50GigaFLOPs


(b) The NVIDIA GTX Titan, released in 2013 with compute power exceeding 1 TeraFLOPs

**Figure 1.10:** The evolution of GPU technology between 1999 and 2013 [29]

above, CPU performance has, however, increased only marginally in that time. This trend is captured in Fig. 1.11 using a performance comparison between Intel CPUs and NVIDIA GPUs over time.




(a) Improvement in floating point operations on CPU and GPU platforms over the last decade

(b) Improvement in memory bandwidth on CPU and GPU platforms over the last decade

**Figure 1.11:** Comparison of improvements in floating point operations and memory bandwidth on CPU and GPU platforms [29]

As can be seen in Fig. 1.11(a), GPUs have delivered better single-precision performance since 2004. Since 2008, GPUs have also been delivering superior double-precision performance. Coupled with the better memory bandwidth offered by this platform (Fig. 1.11(b)), the modern-day GPU has all the necessary attributes to be an attractive alternative to CPUs for the purpose of scientific computing. However, despite its ever-increasing prowess over the last decade, it was not until 2007 that GPU technology was first made accessible to researchers for the purpose of scientific computing. This access came through the public release of CUDA (Compute Unified Device Architecture), a parallel computing

platform and programming model that leverages the powerful compute engine in NVIDA GPUs.

CUDA gives developers access to the virtual instruction set and memory hierarchy of the NVIDIA GPUs. This approach of solving general-purpose (not related graphics rendering) problems on GPUs is known as GPGPU. Using CUDA, a developer is now able to write code that can execute on the GPU using a Single Instruction Multiple Data (SIMD) paradigm. The CUDA platform is now accessible to software developers through CUDA-accelerated libraries, compiler directives (such as OpenACC), and extensions to industry-standard programming languages, including C, C++ and Fortran.

At the core of CUDA lies three important abstractions:

1. A hierarchy of thread groups

2. Shared memory

3. Barrier synchronization

These abstractions are meant to guide the programmer to intuitively partition the problem into coarse sub-problems that can be solved in parallel by 'blocks' of threads, and each sub-problem can be further subdivided into simple tasks that can be solved by all threads within that block. An illustrative example to highlight these three abstractions is presented in the Appendix. For typical computations, several thousands of threads may be spawned over hundreds of cores with each thread mapped to an array index (or an abstract entity such as a pixel,

finite-volume, vortex filament, sediment particle, etc.) and capable of executing a given instruction set. This parallelism is well suited for fine-grain operations where the number of data dependencies per computation is much lower than the size of the problem.

As of June 2013, 54 out of the top 500 supercomputers in the world are built on hybrid systems [30] that are combinations of traditional CPU architectures and GPU or GPU-like processors. The world's second fastest computer is the Titan, shown in Fig. 1.12, a hybrid CPU/GPU based supercomputer built by Cray at Oak Ridge National Laboratory for use in scientific computing. It uses 18,688


**Figure 1.12:** The Titan computer located at the Oak Ridge National Laboratory. As of June 2013, it is the second fastest computer in the world with a theoretical peak of 27 PetaFLOPs [29]

CPUs paired with an equal number of GPUs to perform at a theoretical peak of 27 petaFLOPS. When benchmarked with LINPACK, a peak performance of 17.59 petaFLOPS was observed. With the advent of these hybrid supercomputers, the onus has shifted to the scientific research community to create GPU-accelerated software in order to exploit this compute power or to port existing codes onto GPU platforms. The following section lists a few important studies in the last

five years that have attempted to exploit this technology.

### 1.3.1 Scientific Computing on GPU Platforms

Duraiswami et al [31] presented the first realization of a GPU accelerated Fast Multipole Method (FMM) which demonstrated speedups of 70X on NVIDIA on a GeForce 8800 GTX card (compared to a single CPU core). Hu et al [32] employed the fast multipole method to demonstrate the use of GPUs in speeding up N-body simulations (such as Lagrangian free-vortex methods and particle transport solvers) by over two orders of magnitude.
Graph algorithms have also been successfully ported, with good scalability, onto GPU platforms. Merrill et al [33] demonstrated a GPU-accelerated breadth-first search algorithm that was scalable across multiple GPUs. On standard datasets used for benchmarking, their graph traversal technique was shown to be 26X faster than equivalent CPU computations. Harish et al [34] presented, among other algorithms, a single-source shortest path algorithm that was shown to be 70X faster on an NVIDIA 8800GTX than an equivalent CPU-based search. Vineet et al [35] ported a minimum spanning tree algorithm onto GPUs and demonstrated a 30-50X speedup over equivalent CPU implementations on large datasets ($>$ 5 million nodes).

When GPGPU computing first started gaining popularity, the architectures available were only capable of supporting single-precision computations. This confined the scope of CFD investigations to simulations involving coarse mesh systems where the calculation of flow gradients is not susceptible to floating-

point underflow. As a result, many of the first papers involving GPU-accelerated, mesh-based CFD methods were based on Euler solvers. Hagen et al [36] presented GPU-accelerated simulations of 2D, shock-bubble interactions and 3D, Rayleigh-Taylor instabilities using an Euler solver with an explicit Runge Kutta scheme for time-stepping. Kestener [37] presented a finite-volume Euler solver that demonstrated speedup of 70X on a Tesla S1070 compared to single core CPU computations.

Stock et al [38] presented CPU-GPU hybrid, DNS simulations of an impulsively started sphere at low Reynolds numbers. The authors demonstrated an overall speedup of 22.5X speedup over equivalent CPU calculations. Thibault et al [39] implemented an incompressible Navier-Stokes solver on multi-GPU desktop platforms. As part of this study, the authors showed a speedup of two orders of magnitude over equivalent CPU calculations. Based on the scalability across multiple GPUs seen in this study, the authors suggest the strategy of tackling 'computationally big' problems on GPU clusters with multi-GPUs at each node.

In 2011, Chandar et al [40] unveiled CU++ET, a higher level framework for accelerating CFD codes, developed using OOP techniques available in C++ such as polymorphism, operator overloading and template meta programming. The primary purpose of this framework was to hide the details of GPU computing from the programmer allowing for more focus on the aspects of CFD code development and less on the intricacies of parallelization. In 2012, Chandar et al [41] demonstrated the capabilities of a GPU-based, 3D, unstructured, overset, incompressible Navier-Stokes solver. Speedups of 8-10X were observed in this study.

The authors suggest that the performance gains possible in their implementation are primarily limited by the random nature (non-coalesced) of memory access that is inherent in unstructured solvers.

Stone et al [42] developed a GPU-based library for structured-CFD algorithms including parallel tridiagonal solvers which was then coupled with the CFD solver OVERFLOW. In this study, compute-intensive algorithms such as linear-system inversions were performed on the GPU while the rest of the solver was run on the host. For this reason, the overall performance gain was not significant owing to the overhead of host-device data transfers. Similarly, Jesperson [43] ported the SSOR algorithm implemented inside OVERFLOW onto GPUs. Again, the performance gains observed in this study were not very high because of the need to transfer large matrices between CPU and GPU made this implementation severely limited by host-device memory bandwidth. In contrast to the previous two investigations, the present study seeks to perform all necessary computations on the GPU platform, while using the host only for initialization and file I/O purposes.

Table 1.1 lists the representative speedups demonstrated with different algorithms/implementations on different GPU platforms as presented in the scientific literature.

## 1.4 Objectives

The main objective of this work is to develop a GPU-based, Eulerian-Lagrangian hybrid framework that can be used to study the viscous, turbulent, two-phase

| Algorithm / Implementation | Reference | Year | Platform | Speedup |
|---|---|---|---|---|
| Fast Multipole Method | Duraiswami [31] | 2008 | GeForce 8800 GTX | 70X |
| K-Means Clustering | Zechner [44] | 2009 | GeForce 9600GT | 42X |
| Euler Equations (Unstructured) | Corrigan [45] | 2009 | Tesla C1060 | 33X |
| Incompressible Navier-Stokes | Thibault [39] | 2009 | Tesla S1070 | 100X |
| Boundary Element Method | Stock [38] | 2010 | Tesla S1070 | 22X |
| Euler Equations (Structured) | Kestener [37] | 2010 | Tesla S1070 | 70X |
| RANS (with SSOR algorithm on GPU) | Jespersen [43] | 2010 | GeForce 8800 GTX | 2X |
| Breadth First Search | Merrill [33] | 2011 | Tesla C2050 | 26X |
| Single Source Shortest Path | Harish [34] | 2011 | GeForce 8800 GTX | 70X |
| Vortex Core Extraction | Zhang [46] | 2011 | Quadro 5000 | 25X |
| Traveling Salesman Problem | O'Neil [47] | 2011 | Tesla C2050 | 62X |
| Incompressible Navier-Stokes (with overset meshes) | Chandar [41] | 2012 | Tesla S1070 | 15X |

**Table 1.1:** Representative speedups on different GPU platforms as presented in scientific papers

flowfield associated with brownout conditions. This hybrid methodology will be used to obtain a detailed understanding of the flow physics of scaled, simplified cases such as a hovering MAV scale rotor and an impinging vortex ring. The following are the detailed objectives of the dissertation:

1. Design and develop a GPU-based, finite-volume, multi-granular, compressible, Reynolds Averaged Navier-Stokes solver capable of implicit time stepping.

2. Verification and validation of the solver through the use of canonical test cases in 2D and 3D. These cases will be chosen carefully to verify/validate the various capabilities of the solver as well as to test the limits of the GPU platforms. Some test cases that will be investigated are listed in Table 3.1.

3. The solver will then be benchmarked on available GPU platforms, both gaming and general purpose, to demonstrate the potential of GPU technology to speedup CFD calculations.

4. The GPU-accelerated RANS solver will be coupled to GPU-based Lagrangian methods such as free vortex solvers and particle transport modules, to model brownout conditions. Validation of this framework will be performed by comparing predictions with available experimental data.

## 1.5 Contributions of the Thesis

The key contributions of this research include:

| Index | Test Case | Motivation |
|---|---|---|
| 1 | Shock-Vortex Interaction | Demonstrate speedup for full fine-grain computations |
| 2 | Transonic RAE-2822 | Demonstrate capability to simulate high Reynolds number, 2D turbulent flows with a multi-granular line-implicit time-stepping scheme |
| 3 | ONERA M6 Wing | Demonstrate capability to simulate 3D turbulent flows |
| 4 | 3D Isentropic Vortex | Demonstrate the use of a multi-granular line-implicit reconstruction scheme |
| 5 | Isolated Robin-mod 7 Fuselage | Demonstrate capability of the newest GPU platforms to handle large computational grids |

**Table 1.2:** The test-suite used to verify/validate the GPU-RANS solver

1. The design, development and validation of a GPU-based implicit time-marching compressible RANS solver to simulate brownout conditions.

2. Benchmarking the GPU-RANS solver on various platforms to demonstrate the capability of GPU technology to provide significant speedup over serial computations.

3. Development of a novel FVM-RANS hybrid methodology that combines the numerical efficiency of the free-vortex methods with the high-fidelity of RANS methods to simulate vortical flows similar to those encountered in brownout conditions.

4. Validation of the hybrid methodology by comparing predicted wake structure and velocity profiles with available experimental data.

5. Coupling the hybrid method with a particle-tracking algorithm to simulate

brownout environments. Validation of these dual-phase simulations is performed by comparing predictions of dispersed phase motion with available qualitative experimental data.

6. Performing a detailed analysis of the flow physics involved in laboratory-scale brownout simulations.

## 1.6 Scope and Organization of Thesis

The primary focus in this thesis is on the development of a GPU-based hybrid methodology to model conditions similar to rotorcraft brownout. This serves as an important starting step towards developing computationally efficient predictive tools to suggest mitigation strategies for the complex problem of brownout.

Chapter 2 describes the computational methodology employed in this study. The finite-volume RANS equations are presented in addition to implementation details of the different components of the GPU-based hybrid methodology.

The verification, validation and benchmarking of the GPU-based RANS solver is presented in Chapter 3. The purpose of this chapter is to demonstrate the spectrum of potential speedups that are possible on various GPU platforms for problems of varying size and complexity. In addition, profile data is presented with each test-case to demonstrate the use of GPU resources by each of the participating algorithms.

In Chapter 4, the hybrid methodology is validated by comparing numerical predictions of brownout environments with available qualitative and quantitative experimental data. Two experiments are used to provide validation data:

1. Single-phase flow around an impinging vortex ring [17].
2. Single and dual-phase flow around a hovering, two-bladed micro-scale rotor [16]

For single-phase analysis, the predicted flowfield visualization, vortex trajectories and velocity profiles are compared with experiment to test the applicability of the hybrid methodology to such problems. For dual-phase analysis, the computationally observed mechanisms of transport in the dispersed phase are compared with experimental data to provide qualitative validation for the hybrid method.

Conclusions, other observations noted during the application of the methodologies developed, as well as recommendations for future work are summarized in the final chapter.

# 2

# Methodology

In this chapter, the mathematical description of the flowfield is presented along with a description of the numerical algorithms that make up the hybrid methodology. In addition, the structure and key implementation details of the finite-volume RANS solver is presented including a description of how it is parallelized on GPU platforms. Particular emphasis is given to the optimization of implicit methods for time-stepping and reconstruction on GPUs to maximize performance. The implementation of these algorithms using a multi-granular approach is described in detail.

## 2.1 Flow Domain

One of the key objectives of the current work is to simulate the flowfield in brownout-type conditions, similar to the environment observed around rotors hovering in ground-effect. For this reason, the primary region of interest is chosen to be the zone near the ground plane beneath the momentum source (see

Fig.2.1). The momentum source can be a rotor disk or a jet. To prevent the

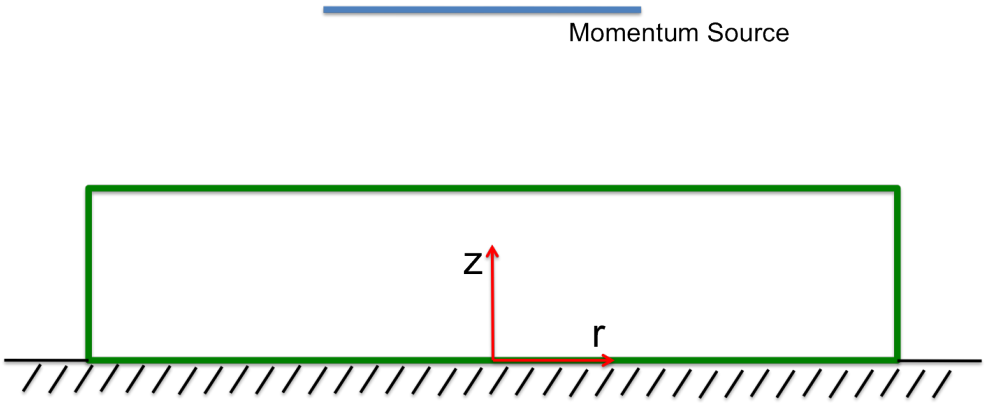



**Figure 2.1:** Primary region of interest for brownout simulations

computational cost from becoming prohibitively large, the radial extent of this zone is limited to a few source (jet or rotor) radii. If the assumption of symmetry is reasonable, further simplification of the flow domain is possible. For instance, the environment around a rotor with an even number of blades can be assumed to be azimuthally symmetric. In this case, only half the domain needs to be modeled. In this domain, the computational model used may be either mesh-based or mesh-free.

### 2.1.1 Mesh-based Models

In mesh-based models, the flow solution is computed only on a finite subset referred to as the 'grid'. The grid is a set of points or control volumes distributed around the domain of interest. In addition to a set of points, the grid also contains connectivity information detailing how the grid points (or cells) are connected to one another. The flow variables represented at each of these grid points constitute the flow solution. If the resolution (number of grid points per unit volume) of

the grid is increased, finer details of the flowfield may be extracted. The meshes used in this method can be 'unstructured' or 'structured' depending on the nature of the connectivity between gridpoints. Examples of mesh-based models are Direct Numerical Simulations (DNS), Reynolds-Averaged Navier-Stokes Solvers (RANS), Euler Solvers, etc. In the present work, a structured RANS solver is the only mesh-based model that is used. Details of this model will be presented later in this chapter.

### 2.1.2 Mesh-free Models

Unlike mesh-based methods that divide the region of interest into spatial subdomains, mesh-free models discretize the flow domain into particle-like entities with physical attributes. These particles are usually free to move and influence other particles depending on the nature of the model used. If inter-particle influences are included in the underlying model, the number of interactions typically scales with the square of the number of particles being simulated. As a consequence, the computational expense increases very rapidly with problem size and this can make high-resolution solutions infeasible. This problem can be circumvented in a sub-class of mesh-free methods through the use of acceleration algorithms such as Fast-Multiple methods, Oct-tree techniques, etc. The use of these methods leads to sub-quadratic performance at the cost of introducing small numerical errors into the solution. Examples of mesh-free models are Smooth Particle Hyrdodynamics (SPH) [48], Free-Vortex Methods (FVM) [49], etc. In the present work, two mesh-free models are used: A filament-based, free-vortex method and a sediment tracking algorithm. Details of these models will

be presented later in this chapter.

## 2.2 Mathematical Description of the Flowfield

The flow-field information is computed by solving the equations of fluid flow, which represent the mathematical statements of three conservation laws of physics:

1. the conservation of mass
2. the conservation of momentum
3. the conservation of energy

These conservation laws can be combined into a single system of partial differential equations called the Navier-Stokes equations. These equations can be numerically discretized and solved with appropriate boundary conditions. To prevent the creation of an under-determined system, additional algebraic or differential equations may be required (e.g. equation of state, Stokes hypothesis or turbulent eddy viscosity equation).

### 2.2.1 Governing Equations

In the present work, the governing equations used in the mesh-based solver are the three-dimensional, unsteady, compressible Navier-Stokes equations that can be expressed in Cartesian coordinates as:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x} + \frac{\partial \mathbf{G}_i}{\partial y} + \frac{\partial \mathbf{H}_i}{\partial z} = \frac{\partial \mathbf{F}_v}{\partial x} + \frac{\partial \mathbf{G}_v}{\partial y} + \frac{\partial \mathbf{H}_v}{\partial z} + \mathbf{S} \qquad (2.1)$$

where $\mathbf{Q}$ is the vector of conserved variables, $\mathbf{F}_i$, $\mathbf{G}_i$, $\mathbf{H}_i$ are vectors representing inviscid fluxes, $\mathbf{F}_v$, $\mathbf{G}_v$, $\mathbf{H}_v$ are vectors that represent the viscous fluxes, and $\mathbf{S}$ represents the body-force source terms. The vectors in the above equations are given by:

$$Q = \left\{ \begin{array}{c} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{array} \right\} \tag{2.2}$$

where $\rho$ is the density, $(u, v, w)$ are the Cartesian velocity components and $e$

is the total energy per unit volume. The inviscid flux vectors are given by:

$$F_i = \left\{ \begin{array}{c} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(e+p) \end{array} \right\} \tag{2.3}$$

$$G_i = \left\{ \begin{array}{c} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ v(e+p) \end{array} \right\} \tag{2.4}$$

$$H_i = \left\{ \begin{array}{c} \rho w \\ \rho w u \\ \rho w v \\ \rho w^2 + p \\ w(e+p) \end{array} \right\} \tag{2.5}$$

The viscous flux vectors are given by:

$$F_v = \left\{ \begin{array}{c} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{array} \right\} \tag{2.6}$$

$$G_v = \begin{Bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} - q_y \end{Bmatrix} \tag{2.7}$$

$$H_v = \begin{Bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} - q_z \end{Bmatrix} \tag{2.8}$$

where $q_x$, $q_y$ and $q_z$ are the thermal conduction terms, which can be represented as a function of temperature according to Fourier's law:

$$q_i = -k\frac{\partial T}{\partial x_i} \tag{2.9}$$

Similarly, the pressure $p$ can be determined in terms of the other primitive variables according to the equation of state for a perfect gas:

$$p = (\gamma - 1)[e - \frac{1}{2}\rho(u^2 + v^2 + w^2)] \tag{2.10}$$

where $\gamma$ is the ratio of the heat capacity at constant pressure ($C_p$) to the heat capacity at constant volume ($C_v$). For air, this ratio is known to be approximately 1.4. The temperature can be expressed as a function of pressure and density:

$$T = \frac{p}{\rho R} \tag{2.11}$$

where R is the gas constant. Using Stokes' hypothesis, the mean stresses can be represented by:

$$\tau_{ij} = \mu[(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_j}) - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}] \tag{2.12}$$

where $\mu$ is the laminar viscosity, which can be evaluated using Sutherland's law.

## 2.3 Non-dimensionalization of Navier-Stokes Equations

To achieve dynamic and energetic similarity for geometrically similar situations and to reduce the number of free parameters, the equations of fluid motion are non-dimensionalized. This non-dimensionalization should provide solutions of such equations that are of the order of one. Typically, a characteristic dimension such as the chord of an airfoil is selected to non-dimensionalize the length scale, while free-stream conditions are used to non-dimensionalize the dependent variables. The non-dimensional variables used in the present work (with superscript $*$) are given below:

$$t^* = \frac{ta_\infty}{L},\ (x^*, y^*, z^*) = \frac{(x, y, z)}{L},\ (u^*, v^*, w^*) = \frac{(u, v, w)}{a_\infty},$$
$$\rho^* = \frac{\rho}{\rho_\infty},\ T^* = \frac{T}{T_\infty},\ p^* = \frac{p}{\rho a_\infty^2},\ e^* = \frac{e}{\rho a_\infty^2},\ \mu^* = \frac{\mu}{\mu_\infty} \quad (2.13)$$

where $L$ is the chord of the airfoil, $a$ is the speed of sound and the subscript $\infty$ represents free-stream conditions.

The non-dimensional parameters are defined as:

$$\text{Reynolds number :} \quad Re_\infty = \frac{\rho_\infty V_\infty L}{\mu_\infty} \tag{2.14}$$

$$\text{Mach number :} \quad M_\infty = \frac{V_\infty}{a_\infty} \tag{2.15}$$

$$\text{Prandtl number :} \quad Pr_\infty = \frac{\mu C_p}{k} \tag{2.16}$$

where $C_p$ is the specific heat at constant pressure. A Prandtl number of 0.72 is assumed in the present work. $V_\infty$ is the free-stream total velocity given by:

$$V_\infty = \sqrt{u_\infty^2 + v_\infty^2 + w_\infty^2} \tag{2.17}$$

The Navier-Stokes equations in non-dimensional form are identical to Eqn. 2.1, if the superscript $*$ is removed. Furthermore, the form of the non-dimensional inviscid and viscous flux terms will also have identical form as before. Differences only arise on closer inspection of the non-dimensional stress and conduction terms, which now appear as functions of Reynolds number and Prandtl number. The non-dimensional mean stresses are given by:

$$\tau_{ij} = \frac{\mu M_\infty}{Re_\infty}[(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_j}) - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}] \tag{2.18}$$

And the non-dimensional thermal conduction terms are:

$$q_i = -\frac{\mu M_\infty}{Re_\infty Pr(\gamma - 1)} \frac{\partial T}{\partial x_i} \tag{2.19}$$

Note that, all the variables in Eqns. 2.18 and 2.19 are non-dimensional. The superscript $\tilde{}$ is deliberately omitted.

## 2.4 Reynolds Averaged Navier-Stokes Equations

The solution of the governing equations, 2.1, do not require any further assumptions in the case of inviscid or laminar flows. However, most realistic flows encountered in engineering practice are turbulent. These flows are characterized by the chaotic, fluctuating motion of fluid molecules, leading to increased momentum and energy transfer between adjacent fluid layers and also at inter-phase boundaries. The accurate resolution of the entire spectrum of fluctuation length-scales requires a very large number of gridpoints. Furthermore, the number of gridpoints required to resolve the flow sufficiently is known to increase super-linearly with Reynolds number. Despite the large leaps made in supercomputer technology, a direct solution (DNS) of Eqn. 2.1 is possible only for problems at the lower end of the Reynolds number spectrum.

One step towards approximating turbulent flows is achieved through the use of the Large- Eddy Simulation (LES) approach. The key observation at the core of LES is that small scales of turbulent motion possess a more universal character than the large scales and can thus be filtered out. Thus, LES seeks to resolve

only the large eddies accurately and to approximate the effects of the smaller scales by using a simple 'subgrid'-scale model. This 'low-pass' filtering leads to LES requiring a significantly smaller number of grid points than DNS, making the investigation of turbulent flows at higher Reynolds numbers more feasible. Despite this benefit, LES is still inherently three-dimensional and unsteady and thus remains computationally expensive. As a result, at the current technology level, LES is not a very good candidate for an engineering tool to study realistic turbulent flows at high Reynolds numbers.

The next level of sophistication is represented by the Reynolds- Averaged Navier-Stokes equations (RANS). This approach begins with the decomposition of the flowfield variables into mean and fluctuating parts. The motivation behind this is that in most engineering applications, only mean quantities are required to be computed with good accuracy. Therefore, any flow variable, $\phi$, can be written as:

$$\phi = \overline{\phi} + \phi' \tag{2.20}$$

where $\phi'$ is the fluctuation and $\overline{\phi}$ is the mean, defined as,

$$\overline{\phi} = \frac{1}{\overline{\chi}} \lim_{\Delta t \to \infty} \frac{1}{\Delta t} \int_0^{\Delta t} \chi\phi(t)dt \tag{2.21}$$

The weighting function $\chi = 1$ for pressure and density and $\chi = \rho$ for all other flow variables (velocity, internal energy, enthalpy and temperature). The RANS equations are obtained by expressing each of the variables in Eqn. 2.1

as a sum of mean and fluctuating parts as in Eqn. 2.20 and assuming that the magnitude of fluctuations are much lower than the mean value of each variable. The resulting system of equations for the mean variables is identical to Eqn. 2.1, with the addition of the Reynolds-stress tensor to the momentum and energy equations that accounts for the additional momentum and energy exchange due to turbulent fluctuations. These stresses add to the viscous stress terms given in Eqn. 2.18 and are given by:

$$\tau_{ij}^{R} = -\rho\overline{u_i' u_j'} \tag{2.22}$$

However, with the introduction of Reynolds-stress terms, we obtain six additional unknowns in the Reynolds-averaged momentum equations. In order to close the RANS equation, the Reynolds stress terms are approximated using a turbulence model. Details of turbulence modeling will be presented in Section 2.6.4.

## 2.5 Initial and Boundary Conditions

The governing equations described in the previous section are general and do not change from one problem to the next. The only distinguishing attribute between different problems is the nature of the initial and boundary conditions used. If these spatial and temporal boundary conditions are applied correctly, a well-posed problem is produced that yields a unique solution.

Initial conditions are specified by assigning the density, flow velocities and pressure at every gridpoint in the flow domain before the start of the computations. One popular approach is the setting of initial conditions that match freestream values. Two common boundary conditions for an external flow are the wall boundary condition and the far-field boundary condition. Wall boundaries are natural boundaries of the physical domain which arise from the wall surfaces being exposed to the flow. In an inviscid simulation, the component of the velocity vector that is normal to the wall surface must be zero. This is referred to as a 'no-penetration' boundary condition. For a viscous fluid which passes a solid wall, the relative velocity between the surface and the fluid at the surface must be zero. This is referred to as a 'no-slip' boundary condition.

While the wall boundary condition corresponds to a realistic boundary, the truncation of the physical domain for the purpose of a numerical simulation leads to the creation of an artificial far-field boundary, where certain physical quantities have to be prescribed. Care should be taken to ensure that the truncation of the domain should not produce spurious effects on the flow solution such as unrealistic acoustic reflections. Additional artificial boundaries can appear at the edges of the flow domain due to the grid topology, model approximations and parallelization technique. For example, the wake-cut boundary appears in C-mesh topologies where grid-points collapse on top of each other. A periodic boundary condition is used when symmetry is invoked to reduce the computational expense of the simulation. For instance, if a 2-bladed hovering rotor is being simulated, the assumption of azimuthal symmetry may be reasonable, allowing the simulation to proceed by only simulating half the domain.

Parallelization of the simulation can also introduce artificial boundaries. If sub-domains of the mesh reside on multiple processors information will need to be transferred between the edges of contiguous sub-domains. For certain kinds of algorithms (particularly when implicit systems are involved), this can lead to the creation of an artificial boundary between adjacent sub-domains which, in turn, can have a significant effect on convergence and accuracy of the simulation. The numerical implementation of both the physical and numerical boundary conditions will be discussed in Section 2.6.6.

## 2.6 GPU-accelerated RANS Solver

The development of a GPU-accelerated, structured-mesh, compressible, unsteady, Reynolds-Averaged Navier-Stokes (RANS) is one of the key contributions of this thesis. The RANS equations are solved using a cell-averaged finite volume formulation described in the following subsection.

### 2.6.1 Finite Volume Formulation

The finite volume technique is a method that directly employs the conservation laws. The Navier-Stokes equations can be expressed in this form by integrating

them over a control volume and applying Stokes theorem:

$$\frac{\partial}{\partial t}\int_{\Omega} Q d\Omega + \oint_{\partial\Omega}(F_i - F_v)dS_x + \oint_{\partial\Omega}(G_i - G_v)dS_y + \oint_{\partial\Omega}(H_i - H_v)dS_z = 0 \tag{2.23}$$

where $\mathbf{dS} = [dS_x, dS_y, dS_z]$ is the surface normal vector and the vectors $Q$, $F_i$, $F_v$, $G_i$, $G_v$, $H_i$, $H_v$ have been defined earlier.

It was first employed by McDonald [50] for the simulation of 2-D inviscid flows. The finite volume method begins by discretizing the governing equations on a number of arbitrary polyhedral control volumes in physical space. For example, a three-dimensional structured mesh will consist of quadrilaterally-faced hexahedrons. A grid cell centered at $(i, j, k)$ has six faces ($N_{faces} = 6$), with face center indices as $(i \pm 1/2, j, k)$, $(i, j \pm 1/2, k)$ and $(i, j, k \pm 1/2)$.

Each surface integral on the left-hand side of Equation (2.23) is then approximated by the sum of the fluxes crossing the individual faces of the control volume, for instance:

$$\oint_{\partial\Omega} F_i dS_x = \sum_{j=0}^{N_{faces}} F_i \Delta S_{xj} \tag{2.24}$$

and the volume integrals are replaced by a zeroth order approximation:

$$\int_{\Omega} Q d\Omega = \overline{Q}\Delta\Omega \tag{2.25}$$

where $\overline{Q}$ is the average value of $Q$ inside each grid cell.

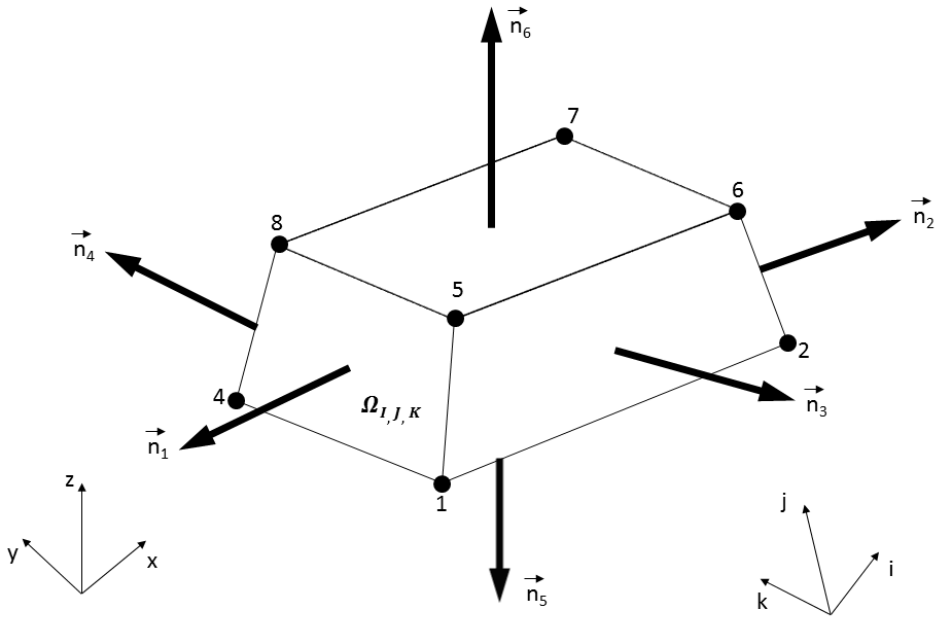



**Figure 2.2:** Representative control volume and face normals for a structured grid in three dimensions

With these approximations, the Navier-Stokes can be written, in discretized form as:

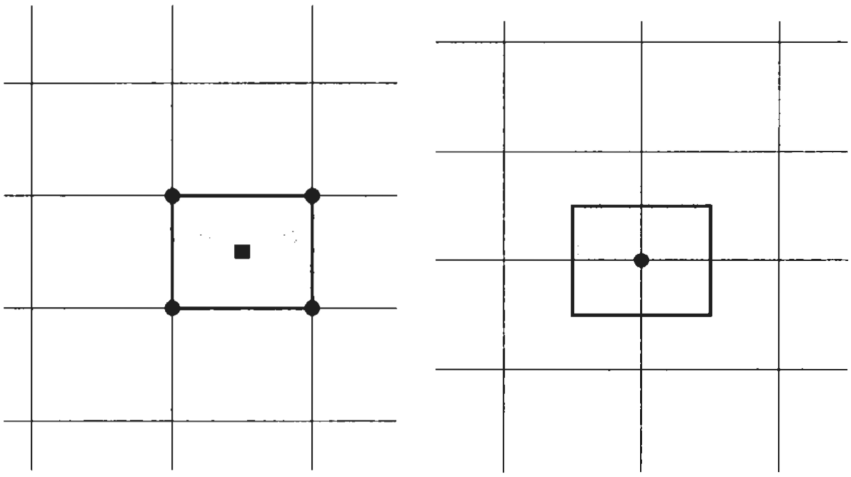
$$\frac{\Delta\overline{Q}}{\Delta t}\Delta\Omega + \sum_{j=0}^{N_{faces}} (F_i - F_v)\Delta S_{xj} + \sum_{j=0}^{N_{faces}} (G_i - G_v)\Delta S_{yj} + \sum_{j=0}^{N_{faces}} (H_i - H_v)\Delta S_{zj} = 0 \tag{2.26}$$

In implementations, there are two popular approaches for defining the shape and position of the control volume with respect to the grid:

1. Cell-centered scheme - here the flow quantities are stored at the centroids of the grid cells. Thus, the control volumes are identical to the grid cells.

Figure 2.3(a) illustrates this.

2. Cell-vertex scheme - here the flow variables are stored at the grid points. The control volume can then either be the union of all cells sharing the grid point, or some volume centred around the grid point (as shown in Fig. 2.3(b)). In the former case we speak of overlapping control volumes, in the second case of dual control volumes.



(a) Cell centered (b) Vertex centered

**Figure 2.3:** Control volumes used in two different finite-volume schemes

The main advantage of the finite volume method is that the spatial discretisation is carried out directly in the physical space. Thus, there are no problems with any transformation between coordinate systems, like in the case of the finite difference method.

### 2.6.2 Inviscid Terms

The inviscid part of the interfacial fluxes (for eg. $F_i$) is computed using upwind schemes. Upwind schemes have the advantage that the wave propagation property of the inviscid equations is taken into account in the flux calculation. The evaluation of the interfacial fluxes involves two steps:

First, the left and right states at each individual interface are computed. The order of accuracy of this evaluation depends on the stencil used (number of points used in the reconstruction). Figure 2.4 illustrates how this is done.

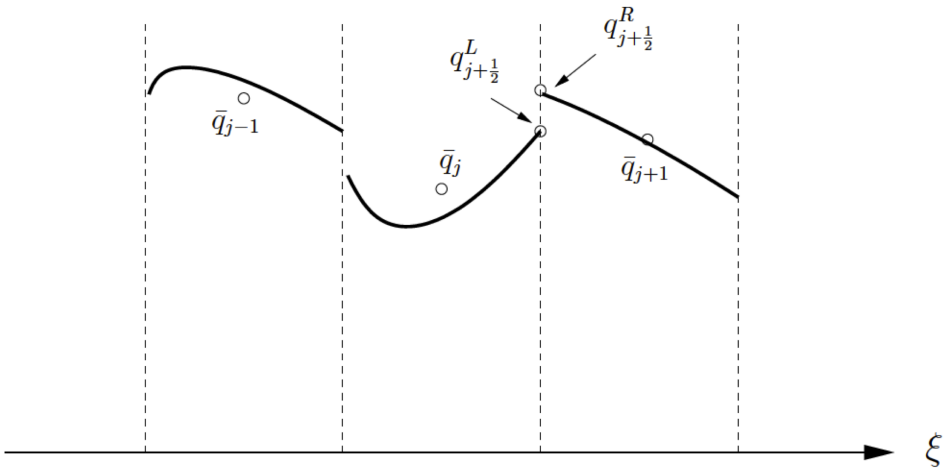



**Figure 2.4:** Schematic showing a one-dimensional reconstruction

As seen in the figure, the left state at interface $j + \frac{1}{2}$ is computed using the reconstruction inside cell $j$ and the right state is computed using the reconstruction inside cell $j + 1$.

The flow solver incorporates three options for reconstruction:

1. Third-order MUSCL scheme with Korens limiter [51]

2. Fifth-order WENO scheme [52]

3. CRWENO [53]

After evaluation of the left and right states at the cell interface, the next step is to calculate the fluxes at the interface. The left and right states can be used to define a local Riemann problem and the interfacial flux can be obtained using any flux splitting scheme. The flow solver uses the Roe flux difference splitting [54] in which, the interfacial flux is given by:

$$\begin{aligned}\hat{\mathbf{F}}_{n,(i+1/2,j,k)} &= \frac{1}{2}(\hat{\mathbf{F}}^{L}_{n,(i+1/2,j,k)} + \hat{\mathbf{F}}^{R}_{n,(i+1/2,j,k)}) \\ &- \frac{1}{2}\left|\hat{A}(\hat{\mathbf{u}}^{L}_{(i+1/2,j,k)}, \hat{\mathbf{u}}^{R}_{(i+1/2,j,k)})\right| (\hat{\mathbf{u}}^{L}_{(i+1/2,j,k)} + \hat{\mathbf{u}}^{R}_{(i+1/2,j,k)})\end{aligned} \tag{2.27}$$

where $\hat{A}$ is the Roe-averaged Jacobian matrix:

$$\left|\hat{A}(\hat{\mathbf{u}}^{L}_{(i+1/2,j,k)}, \hat{\mathbf{u}}^{R}_{(i+1/2,j,k)})\right| = X_{n,(i+1/2,j,k)}|\Lambda_{n,(i+1/2,j,k)}|X^{-1}_{n,(i+1/2,j,k)} \tag{2.28}$$

The subscript $n$ in the right hand side of the above equation signifies that the eigenvalues and eigenvectors for the face are evaluated using the face-normal flow velocities.

Because of the way it is formulated, Roe's flux differencing scheme, as pre-

sented in Eqn. 2.28 will produce an unphysical expansion shock in the case of stationary expansions. Furthermore, the 'carbuncle phenomenon' may be produced, which is a numerical instability that occurs in computations involving multidimensional shock waves.

This deficiency is corrected by an 'entropy correction' suggested by Harten where small eigenvalues of the Roe matrix are modified:

$$|\lambda| = \begin{cases} |\lambda|, & if\ |\lambda| > \delta \\ \frac{\lambda^2+\delta^2}{2\delta}, & if\ |\lambda| \leq \delta \end{cases} \tag{2.29}$$

### 2.6.3 Viscous Terms

The evaluation of viscous stresses involves the calculation of flow variables and their derivatives at cell faces. The flow variables at cell faces are computed using a simple averaging process. If $i$ and $i+1$ are adjacent cells and $i+\frac{1}{2}$ denotes the face between the cells then cell face values are computed using:

$$U_{i+\frac{1}{2}} = \frac{1}{2}(U_i + U_{i+1}) \tag{2.30}$$

where $U$ can be any flow variable.

The remaining task of evaluating first derivatives of velocity components and temperature is performed using second-order, central, finite-differences in curvi-

linear coordinates ($\xi$, $\eta$, $\zeta$) and then transformed to Cartesian coordinates, for eg.,

$$\frac{\partial U}{\partial x} = \frac{\partial U}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial U}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial U}{\partial \zeta}\frac{\partial \zeta}{\partial x} \tag{2.31}$$

### 2.6.4 Turbulence Modeling

The turbulence modeling problem is to close the RANS equation by approximating the Reynolds stress term (Eqn. 2.22). With the assumption of isotropic eddy viscosity, the Reynolds stresses can be represented by:

$$\tau_{ij} = \mu_t[(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_j}) - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}] \tag{2.32}$$

where $\mu_t$ is the turbulent viscosity. The calculation of this turbulent viscosity field is performed by a turbulence model. A host of different turbulence models have been proposed over the last few decades, differing from each other in sophistication and application scope. These models range from zero equation algebraic turbulence models (Baldwin-Lomax [55]), four equation turbulence models ($\overline{v^2}-f$ model [56]) to Reynolds stress models. Reynolds stress models differ from the other popular models in that the eddy viscosity approach is discarded and the Reynolds stresses are computed directly. The zero equation model developed by Baldwin and Lomax calculates the turbulent viscosity as a simple, algebraic function of the conserved quantities. At the other end of the spectrum, the $\overline{v^2} - f$ model by Durbin solves four differential equations to obtain four scalar field vari-

ables. The turbulent viscosity is obtained as an algebraic function of the four solutions to the differential equations.

Even though the zero equation model is computationally less expensive, its applicability is restricted to steady and attached flows [57]. In practice, however, it is a reasonable first approximation even when the conditions of steady, attached flow are not present. Another model that has gained popularity in recent years is the Spalart-Allmaras (SA) model [58]. It was developed specifically with aerospace flow problems in mind. In the present work, the SA model is used in all turbulent computations. The details of this model are presented in the next section.

### 2.6.5 Spalart-Allmaras (SA) Turbulence Model

In the SA model, the Reynolds stresses are related to the mean strain by the isotropic relation, $\overline{u_i' u_j'} = -2\nu_t S_{ij}$, where $\nu_t$ is the turbulent eddy viscosity, which is obtained by solving a PDE for a related variable ($\nu$), given by:

$$\frac{\partial \overline{\nu}}{\partial t} + V.(\nabla \overline{\nu}) = \frac{1}{\sigma}[\nabla.((\overline{(}\nu) + \nu)\nabla \overline{\nu}) + c_{b2}(\nabla \nu)^2] + c_{b1}\overline{S}\overline{\nu} - c_{w1} f_w [\frac{\overline{\nu}}{d}]^2 \tag{2.33}$$

The eddy viscosity $\nu_t$ is related to $\overline{\nu}$ by the relation,

$$\nu_t = \overline{\nu} f_{v1} \tag{2.34}$$

where $f_{v1}$ is a function of $\overline{\nu}$ and the molecular viscosity and is defined as:

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{2.35}$$

where $\chi = \frac{\overline{\nu}}{\nu}$ and $c_{v1} = 7.1$.

The left hand side of Eqn. 2.33 accounts for the convection of the working variable at the mean flow velocity V . The first term on the right hand side represents the diffusion, followed by the production and destruction terms.

Additional definitions are given by the following equations:

$$\overline{S} = \Omega + C_{rot} min(0, S - \Omega) \tag{2.36}$$

where $\Omega$ is the magnitude of the vorticity, $C_{rot} = 2$, $S = \sqrt{2S_{ij}S_{ij}}$ and $S_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$. The constant $C_{rot}$ is an empirical adjustment to ensure a low level of turbulence production in regions such as vortex cores where vorticity exceeds strain rate.

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{2.37}$$

For the destruction term we have:

$$f_w = g[\frac{1+c_{w3}^6}{g^6+c_{w3}^6}]^{\frac{1}{6}} \tag{2.38}$$

$$g = r + c_{w2}(r^6 - r) \tag{2.39}$$

$$r = min[\frac{\overline{\nu}}{\overline{S}\kappa^2 d^2}, 10] \tag{2.40}$$

$$f_{t2} = c_{t3} e^{-c_{t4}\chi^2} \tag{2.41}$$

where the constants are:

$$\begin{aligned} c_{b1} = 0.1355, \sigma = \frac{2}{3}, c_{b2} = 0.622, \kappa = 0.41 \\ c_{w2} = 0.3, c_{w3} = 2.0, c_{v1} = 7.1, c_{t3} = 1.2, c_{t4} = 0.5 \\ c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1+c_{b2}}{\sigma} \end{aligned} \tag{2.42}$$

and $d$ is the distance to the nearest wall.

### 2.6.6 Boundary Conditions

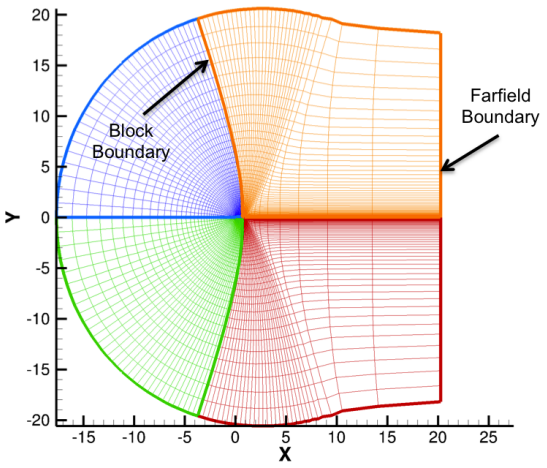
There are several different types of boundary conditions that may be applied on the edges of the domain during the solution of the Navier-Stokes equations. In Section 2.5, a brief description of the physical and numerical boundary conditions that arise during the solution procedure were described. This section describes the numerical treatment of these boundary conditions.

A few typical boundaries are presented on a representative C-mesh around a NACA0012 airfoil in Fig. 2.5.

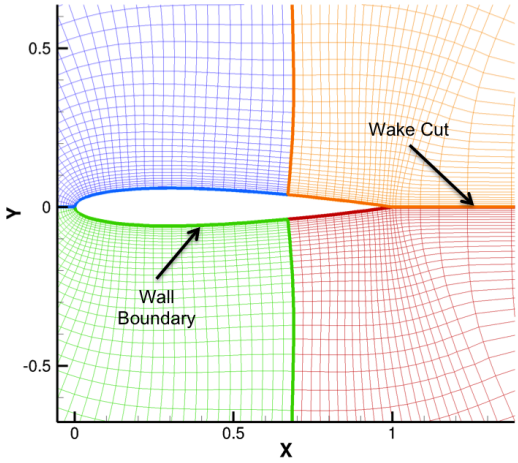
These include a wall boundary, far-field boundaries and wake cut boundary. Another boundary condition that commonly appears in rotorcraft simulations is the periodic boundary. All boundary conditions are implemented through the use of ghost-cells. A layer of ghost cells surround the physical domain. The conservative variables inside these ghost cells are updated depending on the nature of the local boundary condition. A brief numerical description of the implementation details for the different boundary conditions is given below:

#### 2.6.6.1 Wall Boundary Condition

Wall boundary conditions correspond to solid boundaries such as blade or ground surfaces. The wall boundary can be inviscid or viscous. In the finite-volume implementation, these boundary conditions are implemented by prescribing ghost cell values of the conservative variables in a manner that ensures that average quantities at the wall satisfy the no-penetration (inviscid) or no-slip (viscous)




(a)




(b)

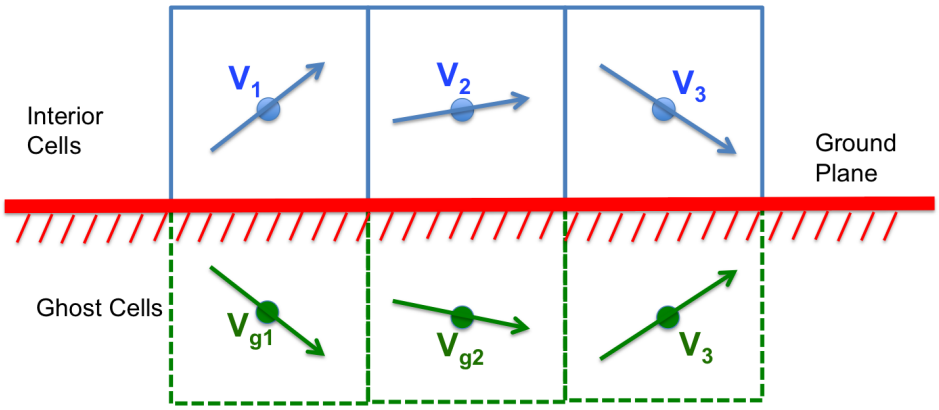
**Figure 2.5:** Boundary conditions typically encountered in simulations

condition. Figure 2.6 illustrates how this is performed.

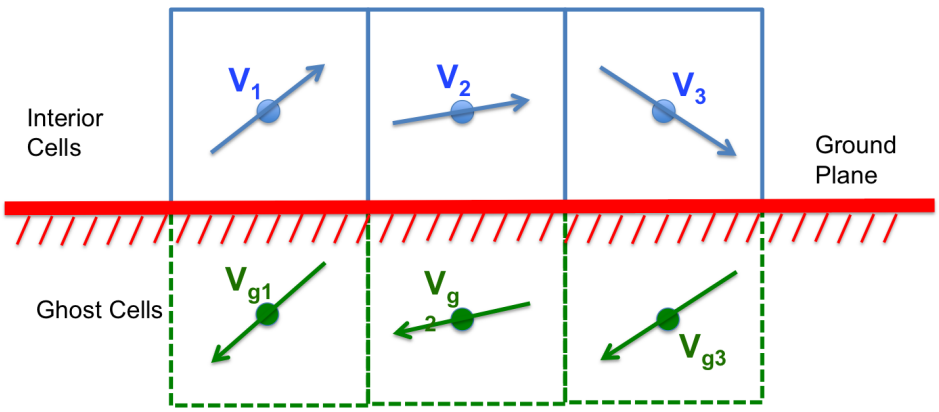
For inviscid simulations, the velocity vector in the ghost cell is set to be the mirror image of the velocity vector in the interior cell adjacent to it. This ensures that the wall-normal velocity component at surface locations (marked as diamonds) goes to zero. A zeroth order extrapolation from the interior cell is performed for density and pressure.

In this work, all the solid walls are treated as viscous walls. Therefore no-slip condition is applied, which requires the fluid velocity at the wall be equal to the zero. For such simulations, the velocity vector in the ghost cell is set to be the negation of the velocity vector in the interior cell adjacent to it. This ensures that the magnitude of the velocity vector at surface locations goes to zero. A zeroth order extrapolation from the interior cell is performed for density and pressure.

In the case of a turbulent simulation, two treatments exist. For a smooth wall, the ghost cell nearest to the wall is given a negative eddy viscosity value which is equal in magnitude to the eddy viscosity in the closest interior cell. This ensures that the turbulent stresses go to zero at the wall. For rough wall simulations, the Boeing wall roughness correction as described in [59] is used. This correction allows for non-zero turbulent stresses at the ground plane.




(a) Inviscid Wall Boundary Condition




(b) Viscous Wall Boundary Condition

**Figure 2.6:** Implementation of the wall boundary condition in the RANS solver

### 2.6.7 Farfield Boundary Condition

Typically, far-field boundaries form the edges of the flow domain that are far removed from body surfaces and large gradients in the conserved quantities to ensure that no spurious wave reflections occur at the boundary. To implement these boundary conditions, the flow is assumed to be locally one-dimensional in a direction orthogonal to the boundary surface and characteristic-based Riemann invariants [60] are used to prescribe the conservative variables in the ghost cells. In this method, based on the direction of the velocity vector and the local sonic velocity, the corresponding Riemann invariants are extrapolated from the interior to the ghost cells (outgoing invariants) or prescribed there based on the free-stream values (incoming invariants).

As mentioned before, the far-field boundaries are typically created far from any body surfaces or momentum source/sinks. As an example, for 2D airfoil cases, the far-field boundary is typically about 20-50 chords away from the solid wall. Analogously, in 3D rotor simulations, the far-field is usually kept about 10 rotor radii away from the rotor center. In order to attain this spatial separation between the region of interest and the far-field, a large number of mesh points are needed leading to a very large computational expense.

The primary objective of the hybrid methodology is the reduction of this spatial separation by avoiding the extension of the domain all the way to freestream conditions. Instead, the boundaries of the RANS mesh are extended upto a locus of points where the flow may be assumed to be largely inviscid. At these points,

the incoming Riemann invariants are set using conservative variables, set, not by freestream values, but by the Lagrangian or analytical component of the hybrid method. Section 2.7.1 describes the process of coupling the two domains. At the farfield, Dirichlet boundary conditions are used for turbulent quantities with free-stream eddy viscosity applied to all such boundaries. It must be noted that this assumes that negligble turbulence is transferred between the rotor plane and the ground - this is an assumption that may need to be revisited at higher Reynolds numbers where the turbulence levels are much higher.

#### 2.6.7.1 Periodic Boundary Condition

In some cases the calculation can be simplified by assuming periodicity along a coordinate direction, leading to a reduction in computational expense. The periodic boundary condition is implemented by copying density and pressure values into the ghost cells at the boundary, and prescribing the velocity vector after performing an appropriate coordinate rotation. Figure 2.7 shows a schematic illustrating the implementation of the periodic boundary condition. The sub-domain $G_1$ consists of ghost cells that get their values from the interior sub-domain $I_1$. Similarly the sub-domain $G_2$ is mapped to $I_2$.

#### 2.6.7.2 Wake Cut Boundary Condition

At the wake cut region, grid planes collapse on to each other. Along these planes, ghost cell values are set equal to the quantities in the interior cells that overlap them. In this sense, the wake-cut boundary condition is very similar to the periodic boundary condition. Wake cuts appear in both C-mesh and O-mesh topologies. Figure 2.8 shows a schematic of a C-mesh topology illustrating the

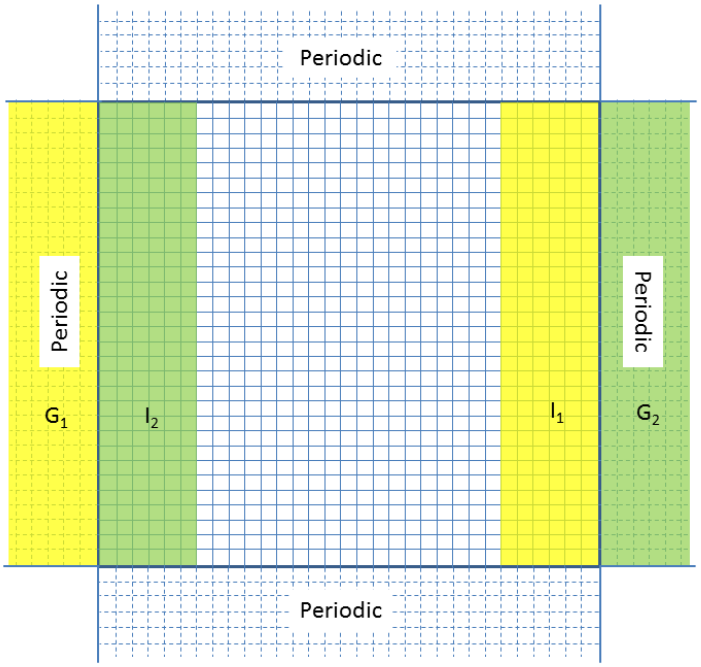



**Figure 2.7:** Schematic illustrating the implementation of the periodic boundary condition in GPU-RANS

implementation of the wake-cut boundary condition. The sub-domain $G_1$ consists of ghost cells that get their values from the interior sub-domain $I_2$. Similarly the sub-domain $G_1$ is mapped to $I_2$.

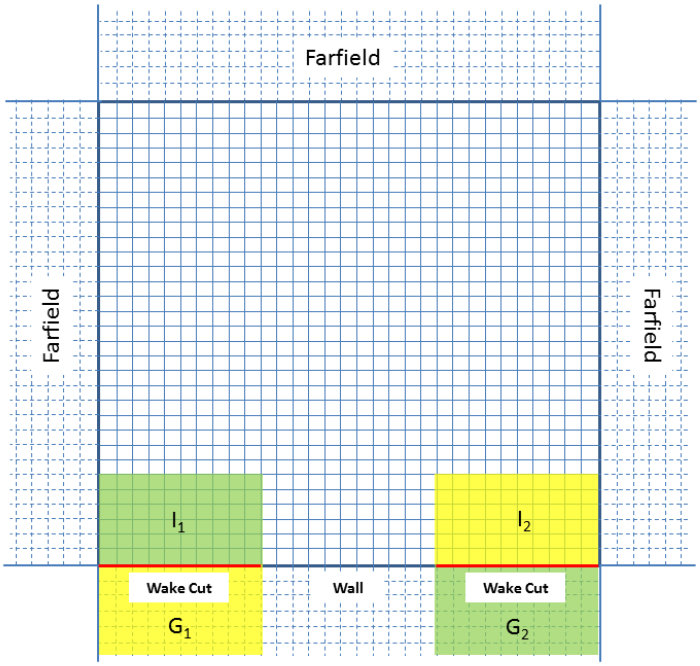



**Figure 2.8:** Schematic illustrating the implementation of the wake-cut boundary condition on a C-mesh topology in GPU-RANS

### 2.6.8 Time Integration

Equation (2.26) can be written in discretized form as:

$$\frac{\Delta \overline{Q}}{\Delta t} \Delta \Omega = -R \tag{2.43}$$

Once the fluxes that contribute to the right hand side of the previous equation are computed, there are two approaches that may be used to update the conservative variables.

#### 2.6.8.1 Explicit Update

The explicit methods only use information at the previous ($n^{th}$) time step to calculate the conservative variables at the new ($(n+1)^{th}$) time step.

$$\frac{\Delta \overline{Q}^n}{\Delta t} \Delta \Omega = -R^n \tag{2.44}$$

While explicit methods are easy to implement, they lead to strict constraints on time step sizes based on the mesh size and flow quantities. The flow solver has the option of using a third-order, multi-stage Runge Kutta scheme for explicit time integration. However, for viscous simulations at high Reynolds numbers that require very fine meshes, the time-step restrictions associated with explicit methods can become prohibitive. Hence, the explicit option is employed only for canonical problems where relatively coarse meshes are used.

#### 2.6.8.2 Implicit Update

Unlike explicit methods, implicit schemes use information at the $(n+1)^{th}$ time-step to perform an update of the conservative variables. Since the state at the $(n+1)^{th}$ time-step is not known apriori, this update requires the solution of a system of equations. While this presents added complexity over the explicit

approach, the advantage of implicit methods is that the time-step constraints are not as strict. For this reason, implict methods are the preferred approach for RANS simulations. For example, the Euler implicit scheme is shown below.

$$\frac{\overline{Q}^{n+1} - \overline{Q}^{n}}{\Delta t}\Delta\Omega = -R^{n+1} \tag{2.45}$$

In addition to the Euler implicit scheme, the solver also has the option of using a higher-order time-integration like BDF2 shown below:

$$\frac{3\overline{Q}^{n+1} - \overline{Q}^{n} + 4\overline{Q}^{n-1}}{2\Delta t}\Delta\Omega = -R^{n+1} \tag{2.46}$$

As mentioned before, the right hand side in Eqn. 2.45 consists of fluxes at time-stage $(n+1)$ which is not known. One technique used is to linearize fluxes about the $n^{th}$ time level and express the $(n+1)^{th}$ state in terms of the state at the previous timestep. A Taylor series expansion is used to perform the linearization.

$$R^{n+1} = R^{n} + \hat{A}\Delta\overline{Q}^{n} + O(h^2) \tag{2.47}$$

where $\hat{A} = \frac{\partial R}{\partial \overline{Q}}$. The linearization shown above is second order accurate.

Substituting Eqn. 2.47 into Eqn. 2.46 and rearranging into the 'delta' form,

$$[I + h\hat{A}]\Delta\overline{Q}^n = -hR^n \tag{2.48}$$

where $h = \frac{\Delta t}{\Delta\Omega}$

The right hand side of the above equation represents the physics of the problem while the left hand side represents the numerics, and consequently determines the convergence characteristics of the system. For structured grids, this system is typically a very large banded set of algebraic equations. While this system is sparse, it is too large for direct inversion to be feasible. For this reason, certain approximations need to be applied to the left-hand side to make the inversion process more efficient, even though this increased efficiency might be accompanied by a degradation in convergence quality. One such approximation is described in the next subsection.

### 2.6.9 Diagonalized Alternating Direction Implicit Algorithm

The Diagonalized Alternating Direction Implicit (DADI) algorithm developed by Pulliam and Chaussee [61] can be used to improve the numerical efficiency of the inversion of system 2.48, the left-hand side of which, can be written in expanded

form as:

$$[I + h(\hat{A} + \hat{B} + \hat{C})]\Delta\overline{Q}^n \tag{2.49}$$

where $\hat{A}$, $\hat{B}$ and $\hat{C}$ are the flux jacobians corresponding to the three different coordinate directions.

The first step in the algorithm is the splitting up of the left-hand side into $N$ factors where $N$ is the dimensionality of the problem.

$$[I + h(\hat{A} + \hat{B} + \hat{C})]\Delta\overline{Q}^n = [I + h\hat{A}][I + h\hat{B}][I + h\hat{C}]\Delta\overline{Q}^n \tag{2.50}$$

The original inversion has now been replaced by three less expensive inversions. The computational expense can be further reduced by the observation that the inviscid components of the flux jacobians can be diagonalized.

$$\begin{aligned} \hat{A} &= T_\xi \Lambda_\xi T_\xi^{-1} \\ \hat{B} &= T_\eta \Lambda_\eta T_\eta^{-1} \\ \hat{C} &= T_\zeta \Lambda_\zeta T_\zeta^{-1} \end{aligned} \tag{2.51}$$

Expressions for $\hat{A}$, $\hat{B}$ and $\hat{C}$ can be found in [61].

Substituting this diagonalized form of the flux jacobians into the discretized Navier-Stokes equation:

$$[T_\xi T_\xi^{-1}+h(T_\xi \Lambda_\xi T_\xi^{-1})][T_\eta T_\eta^{-1}+h(T_\eta \Lambda_\eta T_\eta^{-1})][T_\zeta T_\zeta+h(T_\zeta \Lambda_\zeta T_\zeta^{-1})]\Delta\overline{Q}^n = R^n \tag{2.52}$$

Assuming that the eigenvectors of the inviscid flux jacobians are locally constant, the above equation can be rewritten as:

$$[T_\xi(I+h\Lambda_\xi)T_\xi^{-1}T_\eta(I+h\Lambda_\eta)T_\eta^{-1}T_\zeta(I+h\Lambda_\eta)T_\zeta^{-1}]\Delta Q^n = R^n \tag{2.53}$$

This formulation allows the system to be inverted by performing seven sequential steps:

$$\begin{aligned} S_1 &= T_\xi^{-1}R^n \\ S_2 &= (I+h\delta_\xi\Lambda_\xi)^{-1}S_1 \\ S_3 &= (T_\xi^{-1}T_\eta)^{-1}R^n \\ S_4 &= (I+h\delta_\eta\Lambda_\eta)^{-1}S_3 \\ S_5 &= (T_\eta^{-1}T_\zeta)^{-1}S_4 \\ S_6 &= (I+h\delta_\zeta\Lambda_\zeta)^{-1}S_5 \\ \Delta Q^n &= T_\zeta S_6 \end{aligned} \tag{2.54}$$

The diagonalization step reduces the inversion process to the solution of 5 scalar tridiagonals per coordinate line and 4 matrix vector products per grid cell.

The diagonal algorithm as presented above is really only rigorously valid for the Euler equations. This is because we have neglected the implicit linearization of the viscous fluxes. The viscous flux Jacobians are not simultaneously diagonalizable with the inviscid flux Jacobians and therefore an approximation to the viscous Jacobian eigenvalues have to be used [62] and is given by:

$$\begin{aligned}\lambda_v(\xi) &= \gamma\mu Re^{-1}\rho^{-1}\Omega^{-1} \\ \lambda_v(\eta) &= \gamma\mu Re^{-1}\rho^{-1}\Omega^{-1} \\ \lambda_v(\zeta) &= \gamma\mu Re^{-1}\rho^{-1}\Omega^{-1}\end{aligned} \tag{2.55}$$

$$\begin{aligned}[T_\xi(I + h(\Lambda_\xi - \delta_{\xi\xi}\lambda_v(\xi)))T_\xi^{-1}T_\eta(I + h(\Lambda_\eta - \delta_{\eta\eta}\lambda_v(\eta))) \\ T_\eta^{-1}T_\zeta(I + h(\Lambda_\zeta - \delta_{\zeta\zeta}\lambda_v(\zeta)))T_\zeta^{-1}]\Delta Q^n = R^n\end{aligned} \tag{2.56}$$

The second derivative in the above equation is computed using central differencing.

### 2.6.10 Dual Time-Stepping

Approximation of the LHS results in factorization errors. To remove these factorization errors and to recover time accuracy, one must perform sub-iterations at each physical time step. To carry out these iterations, Eqn. 2.45 can be modified to consider a term that also contains a fictitious pseudo time $\tau$.

$$\frac{\overline{Q}^{k+1} - \overline{Q}^{k}}{\Delta\tau}\Delta\Omega + \frac{\overline{Q}^{k+1} - \overline{Q}^{n}}{\Delta t}\Delta\Omega = -R^{n+1} \tag{2.57}$$

$$[I + h'\hat{A}]\Delta\overline{Q}^{n} = -h'(R^{n} + \frac{\overline{Q}^{k} - \overline{Q}^{n}}{\Delta t}) \tag{2.58}$$

where $h' = \frac{\Delta t}{1+\frac{\Delta t}{\Delta\tau}}$

The above equation can be now be inverted using the DADI algorithm. The unsteady residual at each time-step is now:

$$R^{n} + \frac{\overline{Q}^{k} - \overline{Q}^{n}}{\Delta t} \tag{2.59}$$

During the subiterations, this residual is expected to approach zero. In practice, a drop in residual of the order of one or two orders of magnitude implies that the factorization error has become smaller than the other discretization errors.

## 2.7 Hybrid Methodology

As mentioned in the previous chapter, RANS simulations tend to require a very large number of mesh points. One reason for this large expense is that RANS meshes need to extend out sufficiently far away the region of interest to avoid the creation of unrealistic numerical boundaries. A reduction in the size of the RANS domain would in turn reduce the overall computational expense.

One strategy to reduce the size of the RANS domain is to identify regions of the flowfield where it is required that viscous and turbulent phenomena are properly resolved. In these regions, a high resolution mesh and a RANS solver may be employed for computations. In the remainder of the flowfield a less sophisticated, and consequently more efficient, model may be used.

This approach of domain decomposition with multiple models has been used before in the literature. In [63], a maneuvering helicopter was simulated by using a RANS model for the near-blade regions to capture the 3-D unsteady, transonic flow and dynamic stall. The RANS solver was coupled to a free-vortex method, which was used sufficiently far away from the blade surfaces, to determine the wake structure and the induced flow eld in the vicinity of the rotor blades. The coupling between the two models was performed using a field-velocity method - the velocities induced by the Lagrangian free-wake are imposed upon the CFD-grid in addition to the velocities that are calculated by solving the RANS equations.

One contribution of this thesis is the development of a hybrid methodology, similar to that used in [64] that combines the relatively high-fidelity of a RANS solver with the high computational efficiency of the free-wake method. This hybrid Eulerian-Lagrangian model should be able to overcome the primary disadvantage of the free-vortex method in ground effect i.e. the prediction of unrealistic velocitiy profiles at the ground plane, without incurring the very large computational expense (due to a large number of mesh points) associated with RANS. Furthermore, the proposed method should be able to capture viscous phenomena near the ground such as boundary layer growth and separation, wall-jet formation and vortex-ground interaction.

The first step in the process of developing a hybrid methodology is the identification of regions where viscous/turbulent phenomena are needed to be properly resolved. For a rotor operating in ground-effect, this demarcation is relatively simple. The region near the rotor blades as well as the region near the ground plane would be expected to exhibit viscous/turbulent phenomena such as boundary layer growth and potential separation, vortex-surface interactions, etc. Since we are particularly interested in brownout simulations, special focus is needed on the region near the ground plane where the flowfield interacts with particles. Furthermore, it is assumed that the primary influence between the momentum source and the region of interest is through well-defined vortical structures generated at the momentum source. Since the strength of these vortical structures is determined mostly by integrated attributes such as lift, forcing frequency/strength of the jet, etc., a simplified model such as lifting-line analysis or a prescribed analytical inviscid field can be used near the momentum source.

In the present work, information is transferred between the momentum source and the region of interest using either a free-vortex method (described in detail in Section 2.7.2) or an analytical field (described in Section 2.7.3). The method by which the two interacting solvers in the hybrid method are coupled together is described first.

### 2.7.1 Wake-coupling

Figure 2.9 shows a schematic describing the hybrid simulation as used in ground-effect (IGE) rotor simulations.

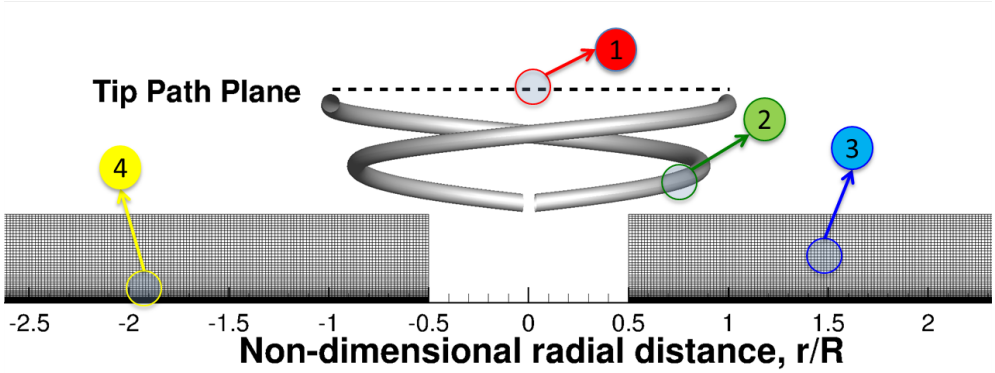



**Figure 2.9:** Schematic describing the structure of the Hybrid FVM-RANS Solver used in an IGE rotor simulation. (1) For the airloads distribution, a linearized aerodynamics module is used. (2) Computations in the far-wake region between the rotor tip path plane are performed using a free-vortex method. (3) A high-fidelity RANS solver is used for computations near the ground plane. (4) For dual-phase flows, the RANS solution is coupled to a particle transport solver

The near-ground computations are done within an Eulerian grid while the far-wake is tracked using the Lagrangian vortex filament model. The airloads at

the rotor disk are computed using a linearized-aerodynamics module and these loads are prescribed to the free-wake solver to determine the tip-vortex strength.

Figure 2.10 shows a schematic describing the hybrid simulation as used in vortex ring simulations. The near-ground computations are done within an Eulerian grid while the far-wake is tracked using the Lagrangian vortex filament model superimposed with an analytical field that describes the axi-symmetric, steady jet.

The strengths and locations of the wake markers, in addition to any existing analytical field, are used to set the incoming characteristic at the outermost CFD grid points. This is achieved in the following way:

1. The current positions and strengths of the Lagrangian markers are used to compute the field velocities. This is performed using Biot-Savart computations. If an analytical field is superimposed upon the Lagrangian markers, the velocities are augmented accordingly. Pressure is extracted from the velocity field using the Bernoulli equation.

2. Density is computed from the pressure field using isentropic relations.

3. The local incoming Riemann invariant is constructed from the calculated primitive variables and set at these outermost grid points.

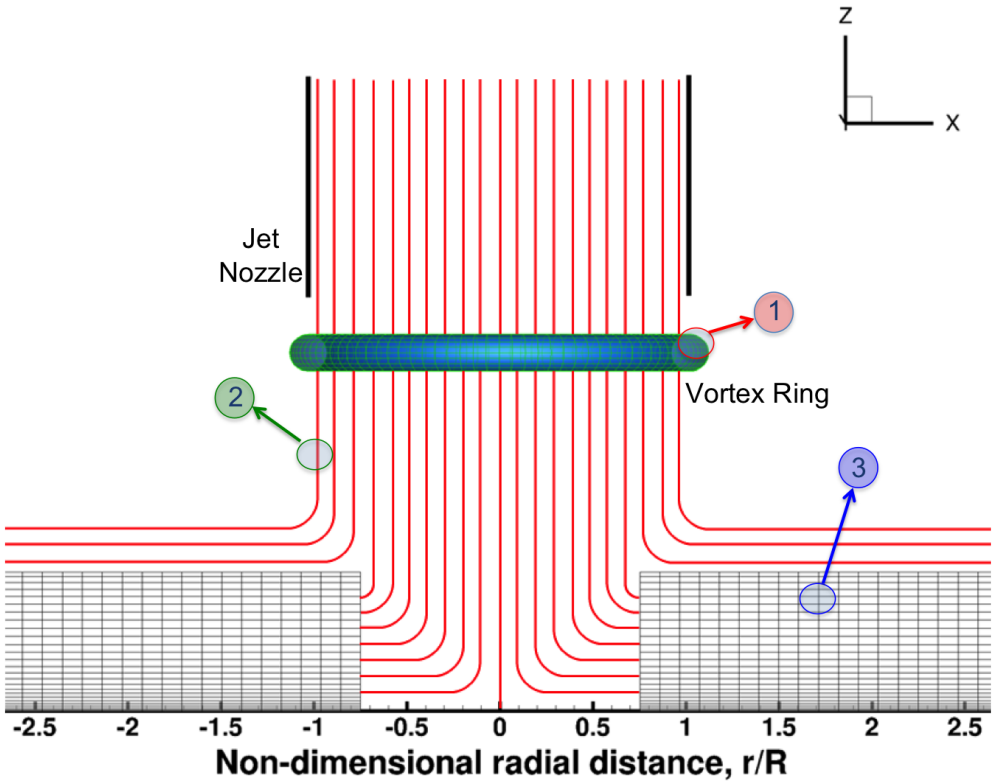



**Figure 2.10:** Schematic describing the structure of the Hybrid FVM-RANS Solver used in the simulation of a vortex ring impinging on a ground plane. (1) For simulating the motion of the ring between the jet lip and the ground plane, a free-vortex method is used with the markers arranged into a ring (2) For simulating the axi-symmetric, steady jet, an analytical field is used to compute the axial and radial velocities (3) A high-fidelity RANS solver is used for computations near the ground plane.

### 2.7.2 FVM Solver for regions away from ground

The free-wake module, PWAM (Parallel Wake Analysis Module), developed at the University of Maryland is a time accurate, efficient, scalable parallel implementation of the solution of the vorticity transport equations in a Lagrangian domain [65]. It was implemented in CUDA to take advantage of the high potential speed up offered by GPU computations. Figure 2.11 shows a snapshot of a FVM simulation of a rotor in ground effect.

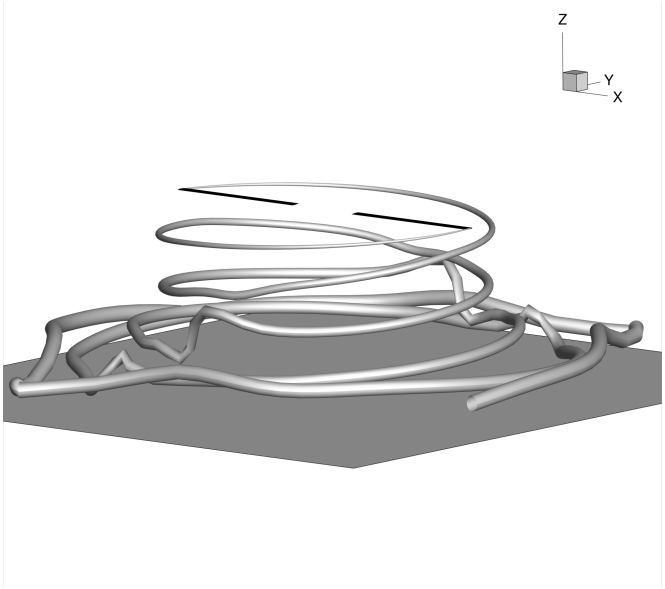



**Figure 2.11:** Filament-based Free Vortex Method used to model the far-wake regions

The wake geometry is discretized into vortex filaments whose strengths are calculated from the aerodynamic forcing provided by a linearized aerodynamics solver. The maximum value of blade bound circulation found in the outer half of the blade is chosen as the filament strength. The convection velocity of each vortex filament is computed by aggregating their mutual influences, the free stream

convection velocity, and an external velocity from a panel code if modeling a fuselage. The mutual influence between the vortex filaments can be computed using the Biot-Savart law. The resulting equations for wake positions are integrated using a fourth-order Runge Kutta time-stepping scheme. The temporal discretization is chosen to be $1^o$ while a $5^o$ discretization is used in wake age. The trailed vortex system consists of vortex filaments released at the tip of each of the two blades and which convect for six revolutions. The spanwise direction was discretized using 20 elements. The near wake region consists of 20 trailed wake filaments spanning over 30 degrees which roll up into the tip vortices. Vortex core growth is modeled using Squire's law and the swirl velocity model is due to Vatistas [66]. The ground plane is modeled using a mirror image of the wake, ensuring no penetration at the ground.

### 2.7.3 Analytical Inviscid Gaussian Jet Field

In [67], Xu et al presented analytical models for inviscid free Gaussian jet solutions. Models for both plane jets and axisymmetric round jets were formulated in this work and calibrated to match experimental data. Expressions and other details of the axisymmetric round jet model are reproduced here.

1. A cylindrical coordinate system is used with $r = 0$ corresponding to the centerline of the jet and $z = 0$ corresponding to the ground plane.

2. The stream function is given by:

$$\psi(r,z) = \frac{-k}{2}(1 - e^{-\frac{r^2}{k}}) + r^2 e^{-\frac{r^2}{k}} \sum_{j=1}^{\infty} c_n L_n^1(\frac{2r^2}{k}) e^{-\sqrt{\frac{8n}{k}}z} \tag{2.60}$$

with

$$\begin{aligned} c_n &= \frac{\int_0^\infty (k/2)(1 - e^{-r^2/k}) L_n^1(2r^2/k) e^{-r^2/k} r dr}{\int_0^\infty r^2 e^{-2r^2/k} [L_n^1(2r^2/k)]^2 r dr} \\ &= \frac{(-1)^n}{n \times n!} \end{aligned} \tag{2.61}$$

where $L_n^1(x)$ are associated Laguerre polynomials of order one. The parameter k is calibrated by comparison with experimental data. It is found to be related to the jet height and radius according to:

$$k = 2.2(a_1 \frac{H^*}{R^*} + b_1) \tag{2.62}$$

where $H^*$ is the distance from the jet to the impinging plate and $R^*$ is the radius of the jet. $a_1$ is a jet shape coefficient, chosen to be 0.07 and $b_1$ is an empirical constant chosen to be 0.294.

3. The radial velocity is given by:

$$u(r,z) = re^{-\frac{r^2}{k}} \sum_{j=1}^{\infty} c_n L_n^1(\frac{2r^2}{k}) \sqrt{\frac{8n}{k}} e^{-\sqrt{\frac{8n}{k}}z} \tag{2.63}$$

4. The axial velocity is:

$$\begin{aligned} w(r,z) &= e^{-\frac{r^2}{k}} + 2(1-\frac{r^2}{k})e^{-\frac{r^2}{k}} \sum_{j=1}^{\infty} c_n L_n^1(\frac{2r^2}{k}) e^{-\sqrt{\frac{8n}{k}}z} \\ &+ re^{-\frac{r^2}{k}} \sum_{j=2}^{\infty} c_n e^{-\sqrt{\frac{8n}{k}}z} \frac{\partial L_n^1(\frac{2r^2}{k})}{\partial r} \end{aligned} \tag{2.64}$$

Figure 2.12 shows the streamlines of a circular impinging jet as predicted by the above model.

### 2.7.4 Modeling Inter-phase and Intra-phase Interactions

The problem of brownout is a very complex dual-phase phenomenon. The modeling of the interactions between and within each phase is described below.

#### 2.7.4.1 Effect of Fluid on Particles

The carrier-phase affects particle motion in two ways:

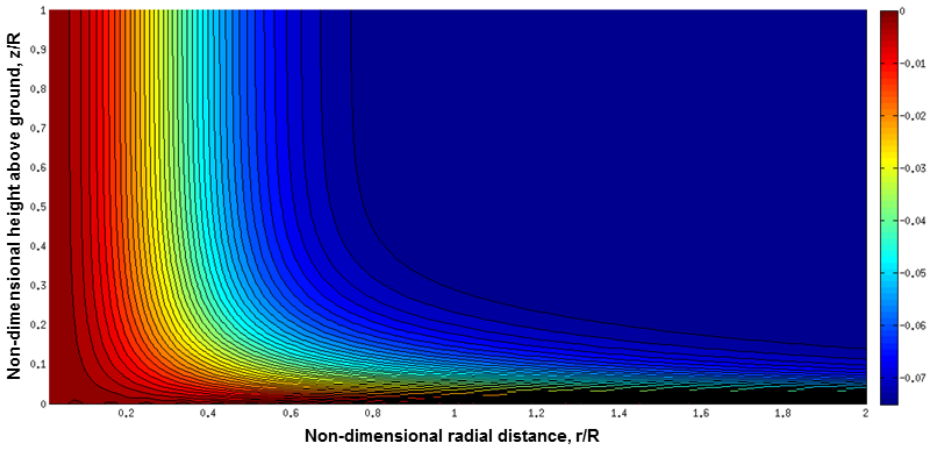



**Figure 2.12:** Streamfunction contours as predicted by the round-jet model proposed by Xu et al

1. Initial entrainment: Local shear stresses in the carrier phase determine whether or not a particle is mobilized.

2. Convection: Once mobilized, the particles move in accordance to the forces applied on it due to the fluid and gravity.

In addition to shear forces, unsteady pressure effects may also play a role in causing particle entrainment. In the present work, these effects are neglected.

#### 2.7.4.2 Effect of Particles on Fluid

The dispersed-phase affects the fluid in two ways:

1. Each particle acts as a sink extracting momentum and energy from the flowfield.

2. The presence of high particle concentrations can modify turbulent quantities in the vicinity leading to a secondary effect on the mean flow.

In the present work, the working assumption is that dual-phase flows similar to brownout are typically dilute. This assumption allows us to decouple the two phases and to model the dual-phase problem as particles moving in a fluid without the fluid feeling the presence of the particles. It is important to note, that while this assumption may be reasonable far away from the ground surfaces, near the boundary layer, the particulate concentrations may be relatively high. In such cases, the assumption of negligible momentum deficit in the carrier-phase due to the presence of particles may lead to inaccurate predictions.

#### 2.7.4.3 Effect of Particles on Particles

Intra-phase interaction in the dispersed phase typically occurs in two ways:

1. Particle-particle collisions - Intersecting trajectories of multiple mobilized particles can cause abrupt changes to convection speeds and direction of motion.

2. Bombardment - High momentum particles can strike the ground plane resulting in the ejection and mobilization of a much larger number of previously stationary particles.

In the present work, particle-particle collisions are neglected under the assumption that, in dilute flows, such interactions are typically rare. Again, like in the case of assuming unidirectional coupling between the phases, this premise is less reasonable near the ground plane where particle concentrations can be relatively high.

The modeling of bombardment is also omitted in the present work, though the reasoning behind this decision is not based on any simplifying assumption and is

instead based on implementation difficulties on GPU platforms. Bombardment occurs when particles impinge upon the ground plane causing several other particles in the 'vicinity' to be ejected. Any implementation of this would require an efficient computation of distances between particles to check which particles are affected by a particlular bombardment event. This reduces to what is essentially a very expensive set of conditionals which, on GPU platforms, can lead to severe performance degradation due to warp divergence.

### 2.7.5 Sediment Tracking Algorithm

The particle tracking algorithm used in this work is developed in Ref. 1. The original work consisted of a Lagrangian method with empirical models for the various phenomena involved in sediment transport - entrainment, convection and bombardment. In the present implementation, the bombardment model has been omitted due to the performance degradation that is expected to accompany it on GPU platforms. Unsteady pressure effects are also neglected. The key models that make up the current implementation of the sediment tracking algorithm are outlined below:

#### 2.7.5.1 Particle Entrainment

Entrainment of particles into the flowfield is modeled using a Bagnold-like model. In this model, the threshold friction velocity at which stationary particles are mobilized is treated as a simple function of particle and fluid density and particle diameter. Figure 2.13 shows the variation of threshold friction velocity as a function of particle size.

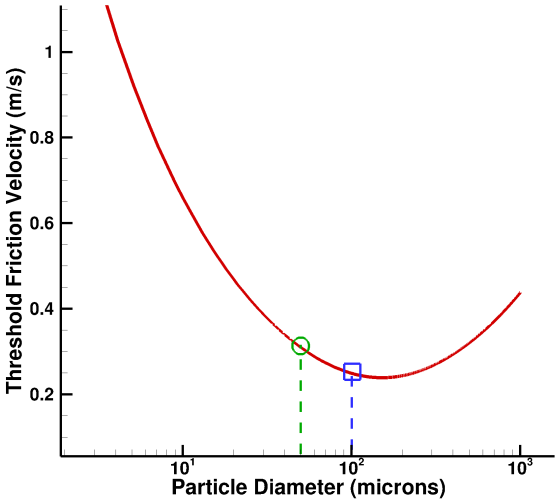



**Figure 2.13:** Variation of threshold friction velocity as a function of particle diameter. The points on the curve corresponding to 100 micron and 50 micron particles are marked in blue and green respectively

It can be observed that at smaller particle diameters, the predicted threshold friction velocity increases with decreasing particle size. This trend is incorporated into the model to account for cohesive forces that vary inversely to particle size. At larger diameters, the increased inertia of the particles leads to larger threshold friction velocities.

The particles are initially located at the ground plane inside the RANS mesh where the carrier-phase velocities are identically zero. The local shear stress at the ground plane is computed and the Bagnold model [68] for entrainment is used to check if this stress exceeds the threshold stress for the particle dimensions being simulated. If the particle is mobilized, it is given a velocity increment, angled at $45^o$ to the horizontal, proportional to the local shear stress and inversely proportional to the mass of the particle. Figure 2.14 illustrates this method. If

the particle manages to escape the boundary layer, it is entrained by the wall jet or a passing vortex. This model for uplifting particles is in contrast to earlier studies where the initial placement of the particles coincided with the edge of the boundary layer where the velocity magnitudes were large enough to entrain the particle directly.

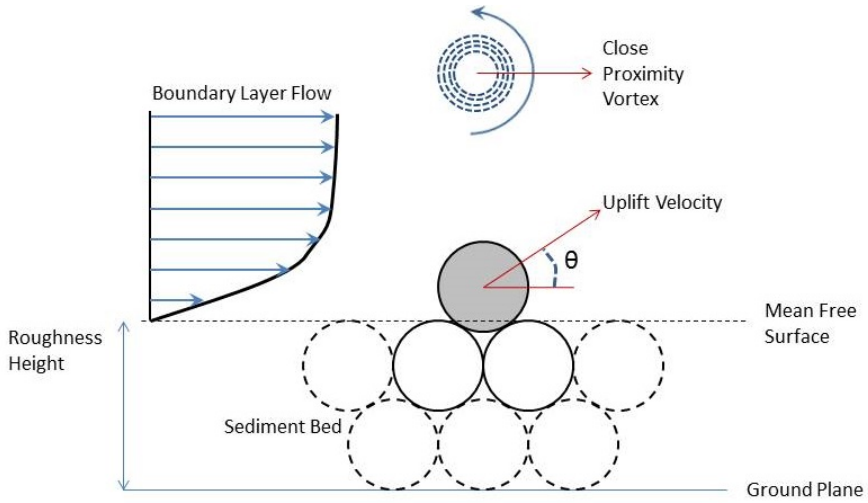



**Figure 2.14:** Modeling the fluid-particle interaction at the ground plane

#### 2.7.5.2 Particle Convection

Once the particles have been entrained into the flowfield, their convection is a direct consequence of the forces acting upon them.

The equation of particle motion is Newton's second law given by:

$$\begin{aligned}
\frac{\partial V_p}{\partial t} &= \Sigma F \\
&= F_g + F_d + F_L + F_{Basset} + F_I \\
&\approx F_g + F_d \\
&= mg + \frac{1}{2}\rho C_d A |V_p - U|(V_p - U) \qquad (2.65)
\end{aligned}$$

In the present implementation, the particles are assumed to be perfect spheres of diameter $d_p$. At low Reynolds numbers (Stokes' Flow), the drag coefficient for a sphere is known to be well approximated by a simple function of Reynolds number:

$$C_d = \frac{24}{Re_p} \qquad (2.66)$$

The equation of particle motion is then reduced to:

$$\frac{\partial V_p}{\partial t} = -\frac{(V_p - U)}{\tau_p} + g \qquad (2.67)$$

where $\tau_p$ is the particle response time:

$$\tau_p = \frac{\rho_p d_p^2}{18\nu} \qquad (2.68)$$

## 2.8 Parallelization

Furthermore, in the present work, every component of the hybrid methodology is run on the GPU platform. This section, therefore, begins with a description of the chief constraints related to GPU computing, followed by details of the various interacting solvers and how they are coupled together.

### 2.8.1 GPU Environment

To optimize for performance on these platforms, the following constraints need to be considered:

1. GPUs are optimized for SIMD operations. For this reason, most of the transistors on the chip are devoted to arithmetic operations and a very small percentage is allocated to handle conditionals. To achieve significant speedup, therefore, care must be taken to ensure that all parallel threads that execute simultaneously (warp) follow the same execution paths. The presence of a conditional causes each branch to be executed serially - a phenomenon referred to as 'warp divergence' - drastically reducing performance. To prevent this, GPU-based code should be designed to minimize or eliminate the number of data-dependent conditionals.

2. When a warp of threads needs to access global memory, an entire bank of contiguous memory is transferred onto the chip. If successive threads need data stored at successive memory locations, this one transaction should be sufficient to satisfy the memory needs of the entire warp. However, if neighboring threads need to access memory locations that are separated by

large strides (non-coalesced memory accesses), separate memory transactions are performed per thread leading to a reduction in performance. In the worst case, each thread in the warp will need one separate transaction with VRAM leading to 32 transactions per warp which can lead to a significant reduction in performance. One strategy to mitigate this issue is to transfer relevant data to shared or register memory to allow for low-latency reuse of this memory. Another strategy is to order your data structures in a way to ensure that successive threads are accessing contiguous memory locations as much as possible.

3. The memory bandwidth between host and device is usually much lower than the bandwidth between VRAM and device. For this reason, any GPU-based solver should be implemented in a way that minimizes or eliminates host-device transfers.

In the context of GPU-based solvers, two modes of parallelism may be exposed:

1. Fine-grain parallelism - Operations that can be performed independently on each abstract element (pixel, cell volume, filament, particle, etc.) are tackled by assigning each such element to a different GPU thread to maximize parallel efficiency. This is similar to how graphics rendering is handled on a GPU with each thread being mapped to a separate thread. Figure 2.15 illustrates the nature of fine-grain computations. Algorithms such as explicit reconstruction schemes, inviscid/viscous flux calculation and filament/particle convection are inherently data parallel and therefore can be processed in a fine-grain manner.

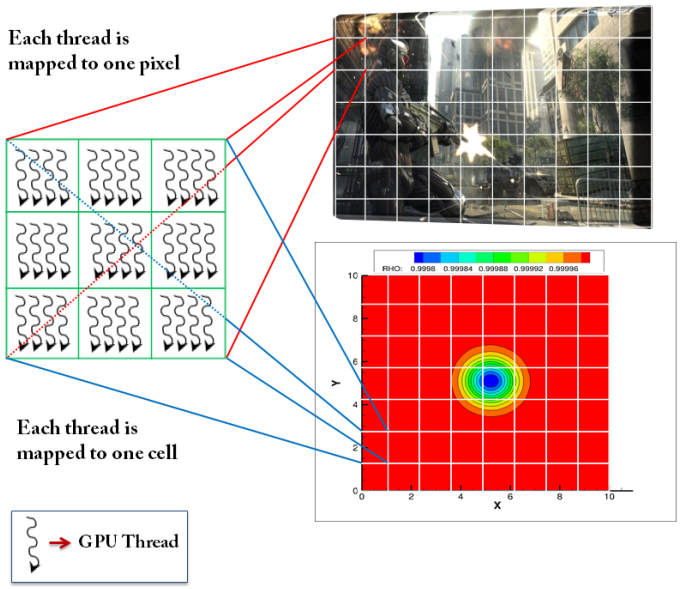



**Figure 2.15:** Fine grain parallelism employed in the RANS solver. These operations are what the GPU architecture is optimized for: graphics rendering is performed by assigning the task of reshading each pixel to a separate thread. Similarly, in the RANS solver, data-parallel operations are performed by mapping each thread to a separate finite-volume

2. Coarse-grain parallelism - Implicit schemes, such as those involved in algorithms for reconstruction or time-stepping, require the creation and inversion of a linear system composing of multiple cell volumes. For instance, ADI methods in grid-based CFD methods generate linear systems that couple cell quantities along a coordinate line. Figure 2.16 illustrates the structure of coarse-grain computations for algorithms such as ADI. In such cases, a coarse grain strategy is pursued assigning each line to a different GPU thread. Similarly, compact schemes, such as CRWENO [53], involve a linear coupling of cell quantities along coordinate lines to solve for reconstructed values at cell interfaces. These schemes can also be tackled by mapping each implicit system to a separate GPU thread. While this strategy does result

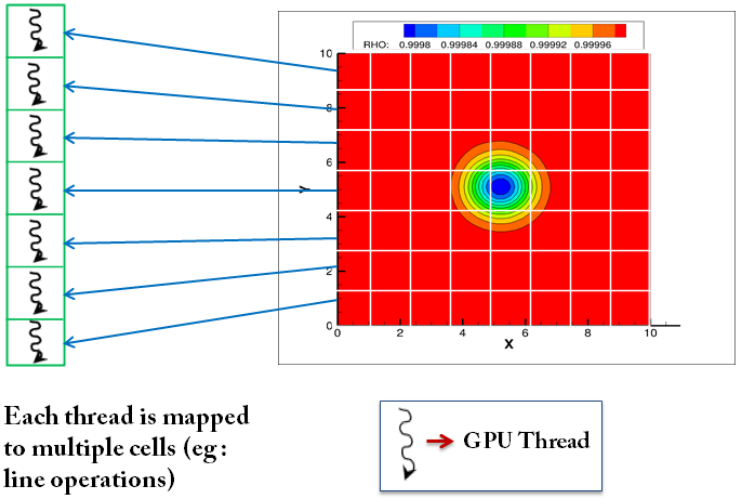



**Figure 2.16:** Coarse grain parallelism employed in the RANS solver. This mode is employed during the inversion of sparse linear systems such as those encountered in implicit time-stepping/reconstruction routines. In these cases, an entire linear system is mapped to a single thread.

in a performance penalty due to loss of fine-grain parallelism, the accompanying benefits of using an implicit scheme (increased stability, high-order compact schemes) make the additional work worthwhile. For instance, use of an implicit scheme for time-stepping, frees the time-step size from the Courant-Friedrichs-Lewy stability constraint.

### 2.8.2 RANS Solver

Figure 2.17 illustrates the structure of the RANS solver. A large number of arrows indicate that the associated routine is computed in a fine-grain manner while a lower number of arrows indicate the use of a coarse-grain approach. Only the procedures colored red are performed on the host. All other routines are performed on the device.

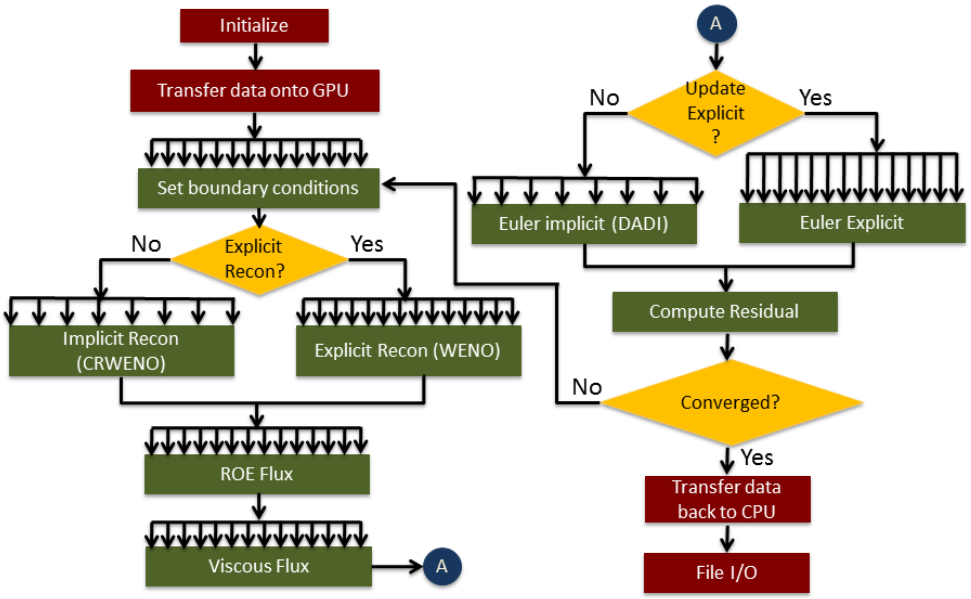



**Figure 2.17:** Flowchart for the GPU-RANS solver

In many situations, a combination of fine and coarse-grain parallelism can be used within the implementation of a single algorithm. This multi-granular approach can result in significant speedup, particularly in the solution of implicit systems. For instance, certain families of implicit methods construct linear systems that only couple cell quantities along a coordinate line. For example, Alternating Direction Implicit (ADI) schemes split the implicit operation into $D$ consecutive line operations where $D$ is the dimensionality of the problem. This approximate factorization incurs an error which can be reduced by performing subiterations at each timestep. For steady test-cases, since the transients are not important, the use of subiterations can be avoided during convergence. ADI methods are suitable for SIMD architectures such as GPUs because each line operation can be performed independent of all other lines in the same coordinate

direction. This inherent line-parallelism in ADI methods allows for a coarse grain strategy to be pursued by assigning each line to a different GPU thread for the purpose of setting up and inverting each linear system. In preliminary studies, this approach was used for algorithms such as the Diagonalized Alternating Direction Implicit (DADI) method (Ref. [61]). However, it was found that the accompanying performance penalty resulted in the implicit time-stepping scheme being substantially slower than an explicit time-stepping scheme such as TVD-RK3. A closer look at this strategy suggests scope for further improvement. Consider, for instance, the DADI algorithm in two dimensions:

$$[T_\xi(I + h\delta_\xi\Lambda_\xi)T_\xi^{-1}T_\eta(I + h\delta_\eta\Lambda_\eta)T_\eta^{-1}]\Delta Q^n = R^n \tag{2.69}$$

This formulation allows the system to inverted by performing five sequential steps:

$$S_1 = T_\xi^{-1}R^n \tag{2.70a}$$

$$S_2 = (I + h\delta_\xi\Lambda_\xi)^{-1}S_1 \tag{2.70b}$$

$$S_3 = (T_\xi^{-1}T_\eta)^{-1}R^n \tag{2.70c}$$

$$S_4 = (I + h\delta_\eta\Lambda_\eta)^{-1}S_3 \tag{2.70d}$$

$$\Delta Q^n = T_\eta S_4 \tag{2.70e}$$

Expressions for $T_k$ and $\Lambda_k$ can be found in [61].

Compared to the non-diagonalized form, this scheme reduces the number of floating point operations by approximately 75% which allows for more efficient

serial computations, which was the primary motivation behind the algorithm's design. The structure of this scheme, however, also makes it very well suited for achieving significant speedup on GPU platforms without exceeding memory constraints. This is made clear by considering the two important steps involved in the derivation of the DADI scheme:

1. Approximate factorization

2. Diagonalization of the flux jacobians and the assumption of locally constant eigenvector matrices

Each of these steps play key roles in making the DADI scheme a prime candidate for LHS inversion on GPUs:

1. The approximate factorization of the original system decouples the equations across the different coordinate directions. This effectively reduces a large block pentadiagonal to a series of smaller, line-parallel, block tridiagonal systems. A reduction in linear system bandwidth reduces the memory requirements required to compute and invert it.

2. The diagonalization step coupled with the assumption of locally constant eigenvectors further reduces memory requirements by removing the need to store a minimum of three, dense, flux jacobian matrices per grid cell. In addition to the two eigenvector matrices (assumed to be locally constant), only the diagonal elements of the three blocks corresponding to the local eigenvalues need to be computed and stored.

3. The assumption of locally constant eigenvalues reduces a block triadiagonal

system to a scalar tridiagonal system which requires less computational expense to invert.

4. Diagonalization also decouples the equations across the conservative variables providing one additional dimension along which to parallelize the implicit system one is solving. For instance, along a line, one thread can be tasked with handling the update of density, another thread can handle the update of x-momentum and so on. This allows for more threads to be spawned during a given timestep which, in turn, can lead to the GPU cores being better utilized. In addition, if the data structures used to hold the conservative variables are ordered so that the variable dimension is mapped to the most efficient (unit) stride in memory (as shown in Fig. 2.18), this parallelism ensures an upper-bound on the number of VRAM-to-device transactions per half-warp of threads for most operations ( 3 for 3D RANS computations).

   The parallelization of the variable dimension can significantly improve efficiency and GPU utilization. Figure 2.19 illustrates variable-dimension parallelism.

   This parallelization is even more effective in 3D than it is in 2D. To illustrate this, consider the following example: A coarse-grain operation on a 2D cartesian mesh with dimensions IMAX,JMAX is illustrated in Fig. 2.20(a).

   A total of IMAX systems, each of size JMAX, is shown being solved in parallel. As an example, if IMAX=20, this will require the utilization of only 20 cores and most of the remaining cores on the GPU will be rendered idle during this operation (The GTX Titan has a total of 2688 cores). Variable

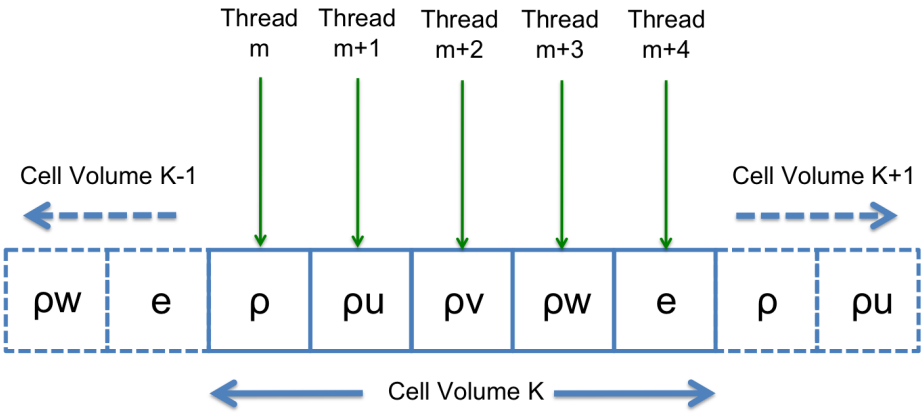



**Figure 2.18:** Ordering of data structure to ensure that the variable dimension is mapped to the most efficient (unit) stride in memory. For both fine-grain and coarse-grain operations, this reduces the number of transactions with VRAM per warp of threads

parallelism in 2D (with four conserved variables), as shown in Figs. 2.19 2.18, will increase the total number of concurrent threads to 80.

In contrast, Fig. 2.20(b) shows a representative coarse-grain operation on a 3D cartesian mesh with dimensions IMAX,JMAX,KMAX. A total of IMAX * JMAX systems, each of size KMAX, is shown being solved in parallel.

If IMAX=20 and JMAX=20 this line-parallelism will lead to the utilization of 400 cores. With variable parallelism added in, the total number of concurrent threads is increased to 2000 cores (RANS computations involves 5 conserved quantities in 3D). Figure 2.21 illustrates this combined line/variable-parallelism in 3D.

On a GTX Titan, this line/variable-parallelism will utilize close to 75% of

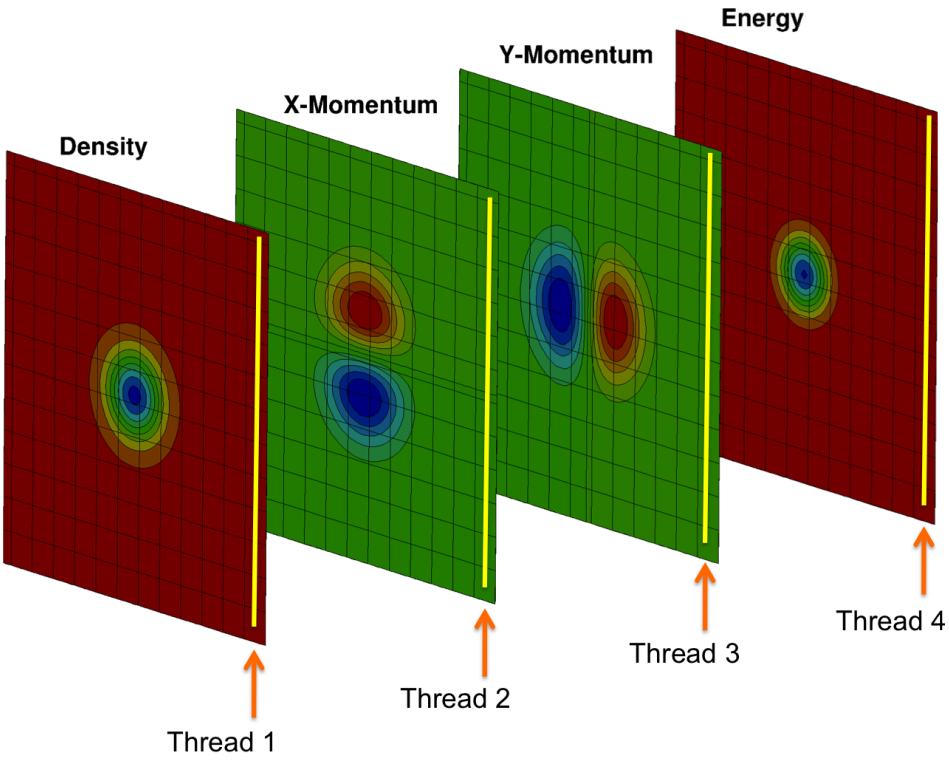
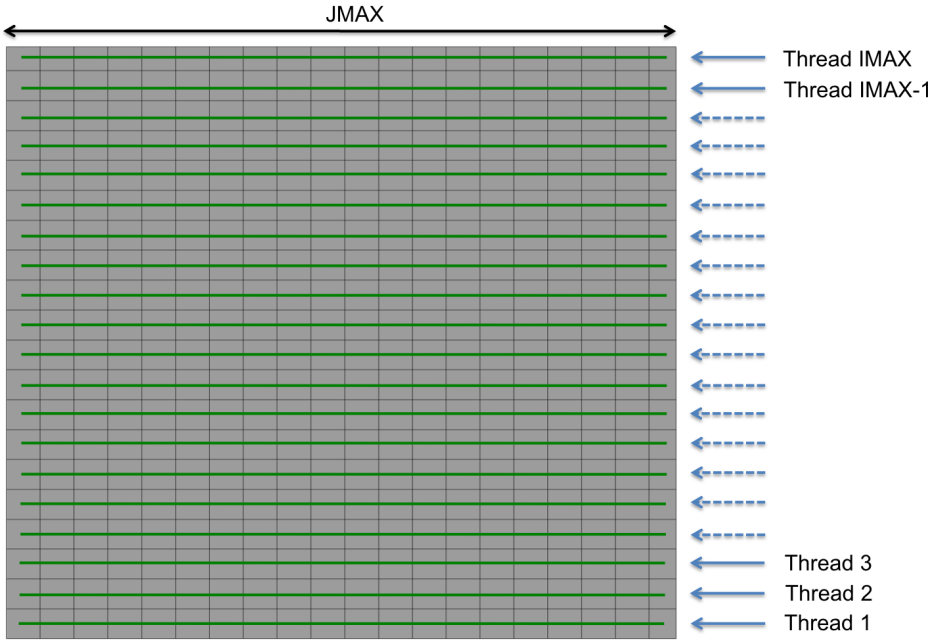
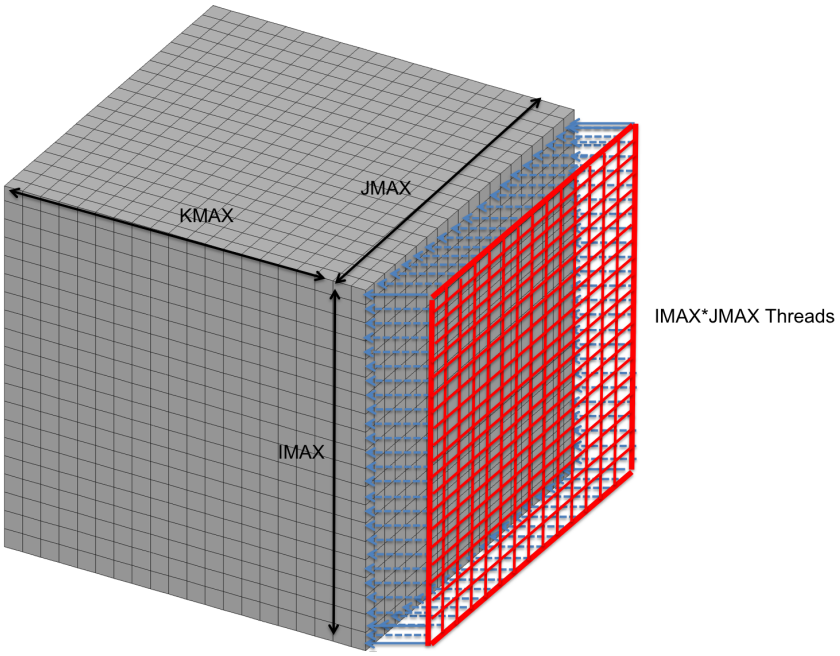



**Figure 2.19:** Parallelization of the variable dimension allows for coarse-grain operations to be more efficient. Firstly, a larger number of threads can be spawned for a given operation leading to better utilization of the GPU. If the implementation includes a data-structure with variables along the most efficient stride, successive threads are often operating on contiguous memory locations during the setting up of implicit systems. This reduces the number of VRAM transactions and increases speedup.




(a) Line parallelism in 2D RANS computations




(b) Line parallelism in 3D RANS computations

**Figure 2.20:** Line parallelism in 2D and 3D RANS computations

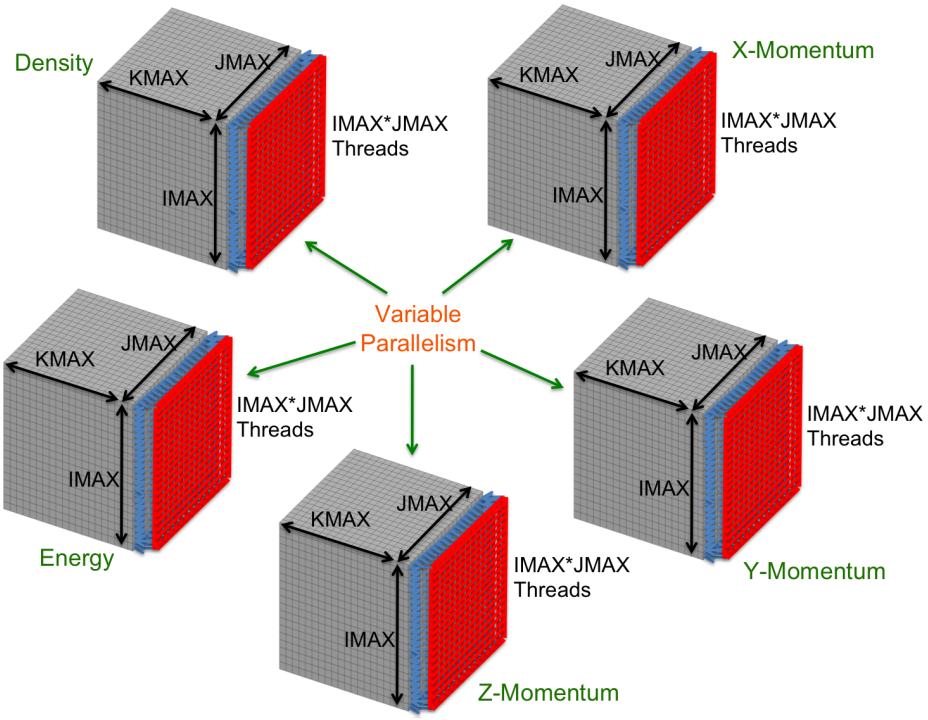



**Figure 2.21:** Line/Variable parallelism in 3D RANS computations

the GPU resources during the coarse-grain operation while the equivalent 2D example presented earlier uses only 3% of the GPU resources. Parallel efficiency is clearly superior in 3D for RANS computations.

5. The combination of approximate factorization and diagonalization allows for the use of a multi-granular approach to invert the LHS. Equations 2.70(a), 2.70(c) and 2.70(e) are matrix-vector multiplications that can be performed independently at each cell. Hence, for these operations, the solver can revert to fine-grain parallelism. Equations 2.70(b) and 2.70(d) consist of four scalar tridiagonal inversions each. For the inversion itself, the traditional Thomas Algorithm is employed and is not suitable for fine-grain parallelism. However, the setting up of each tridiagonal is largely data-

parallel. So, for steps 2.70(b) and 2.70(d), the solver first constructs the implicit system in fine-grain mode and then inverts them in coarse-grain. It is found that this approach of using multi-level granularity results in further significant savings.

Similarly, in the case of implicit reconstruction, this strategy of using two levels of parallelism is found to give the best results. Consider the CRWENO algorithm for reconstruction at cell interfaces (Ref. [53]):

$$\begin{aligned}\frac{2\omega_1+\omega_2}{3}f'_{j-1/2}+\frac{\omega_1+2(\omega_2+\omega_3)+\omega_4}{3}f'_{j+1/2}+\frac{\omega_3+2\omega_4}{3}f'_{j+3/2}=\\ \frac{\omega_1}{6}f_{j-1}+\frac{5(\omega_1+\omega_2)+\omega_3}{6}f_j+\frac{\omega_2+5(\omega_3+\omega_4)}{6}f_{j+1}+\frac{\omega_4}{6}f_{j+2}\end{aligned} \tag{2.71}$$

This is a tridiagonal system that couples interface values along a coordinate line. The weights in Eqn. 2.71 are dependent only on local quantities as can be seen in Eqns. 2.72(c) - 2.72(f).

$$\omega_k = \frac{\alpha_k}{\Sigma_k \alpha_k} \tag{2.72a}$$

$$\alpha_k = \frac{c_k}{(\beta_k+\epsilon)^m} \tag{2.72b}$$

$$\beta_1 = \frac{13}{12}(f_{j-2}-2f_{j-1}+f_j)^2+\frac{1}{4}(f_{j-2}-4f_{j-1}+3f_j)^2 \tag{2.72c}$$

$$\beta_2 = \frac{13}{12}(f_{j-1}-2f_j+f_{j+1})^2+\frac{1}{4}(f_{j-1}-f_{j+1})^2 \tag{2.72d}$$

$$\beta_3 = \frac{13}{12}(f_j-2f_{j+1}+f_{j+2})^2+\frac{1}{4}(3f_j-4f_{j+1}+f_{j+2})^2 \tag{2.72e}$$

$$\beta_4 = \frac{13}{12}(f_{j+1}-2f_{j+2}+f_{j+3})^2+\frac{1}{4}(-5f_{j-2}+8f_{j+2}-3f_{j+3})^2 \tag{2.72f}$$

Therefore, similar to the approach used in the DADI algorithm, the tridiagonal entries can be computed in fine-grain mode with only the actual inversion (using the Thomas algorithm) being performed in coarse-grain.

### 2.8.3 Free Wake Solver

Similar to the RANS component of the hybrid methodology, the free-wake solver is also implemented in CUDA to be able to run on GPU platforms. This also enables the entire hybrid simulation to be run on the GPU without having to transfer information back and forth between the host and device. The most computationally expensive part of the free-wake solver is the computation of induced velocities on the vortex filaments. For these computations, a fine-grain parallelism approach is used with each GPU thread mapped to a single vortex filament and responsible for accumulating induced velocities due to the other filaments.

### 2.8.4 Sediment Tracking Algorithm

Similar to the case of the free-wake solver, a fine-grain parallelism approach is used with each GPU thread mapped to a single particle and responsible for interpolating local flow velocities and performing sediment convection.

## 2.9 Summary

In this chapter, the computational methodology used in the hybrid solver was presented in detail. Further, the governing equations and structure of the GPU-based

RANS solver were presented along with key implementation techniques. Since implicit methods are typically expensive bottlenecks in RANS simulations, the multi-granular optimization of important implicit algorithms (DADI, CRWENO) was presented. Finally, strategies to avoid performance degrading pitfalls (warp divergence, increased VRAM transactions) on GPU platforms were suggested.

# 3

# Verification and Validation

Before using the hybrid methodology to simulate rotorcraft brownout, it is necessary first to ensure the proper working of each component of the methodology to gain confidence in the solution algorithm. Both the free-wake solver and the sediment tracking algorithm used in the current work have been tested extensively in other studies [1, 19, 20, 69]. The current chapter is therefore focused on the issue of verification, validation and benchmarking of the third component of the hybrid methodology - the GPU-accelerated Navier-Stokes solver. A test-case suite is carefully chosen to highlight the various capabilities of the solver as well as to compare performances on various GPU platforms when certain algorithms, particularly those embodying a coarse-grain parallelism, are used. The test-suite that is used in the present verification/validation study is shown in Table 3.1

In each case, first a validation or verification process is performed by comparing solver output with experimental data or an exact solution. In addition, a performance study is conducted on various GPU platforms and compared with equivalent serial computations on a single CPU core.

| Index | Test Case | Motivation |
|---|---|---|
| 1 | Shock-Vortex Interaction | Demonstrate speedup for full fine-grain computations |
| 2 | Transonic RAE-2822 | Demonstrate capability to simulate high Reynolds number, 2D turbulent flows with a multi-granular line-implicit time-stepping scheme |
| 3 | ONERA M6 Wing | Demonstrate capability to simulate 3D turbulent flows |
| 4 | 3D Isentropic Vortex | Demonstrate the use of a multi-granular line-implicit reconstruction scheme |
| 5 | Isolated Robin-mod 7 Fuselage | Demonstrate capability of the newest GPU platforms to handle large computational grids |

**Table 3.1:** The test-suite used to verify/validate the GPU-RANS solver

In each test-case, the solver is run in serial (with -O2 optimization) and on various NVIDIA GPU platforms. All computations on these GPUs were performed at double precision. Details about these platforms are listed in Table 3.2. Serial computations were performed on a single Intel i5 core with a clock speed of 3.1GHz.

| GPU Type | Cores | Memory (MB) | Clock (MHz) | Memory Bandwidth (Gb/s) |
|---|---|---|---|---|
| GTX640 | 384 | 2078 | 797 | 28.5 |
| GTX480 | 480 | 1536 | 700 | 177.4 |
| GTX580 | 512 | 1536 | 772 | 192.4 |
| Titan | 2688 | 6144 | 837 | 288 |

**Table 3.2:** Description of the different GPU platforms used in simulations

Lastly, profile data is extracted to study the distribution of resources among the various participating algorithms. All profile data shown in this chapter neglects the resources spent on transferring data to and from the device and focuses

instead on the resources used for computations. Transferring data between the host and the device is expensive, but in most cases, this transfer occurs infrequently during a simulation - once during initialization on the device and once during every I/O operation.

All test cases that follow use a grid of thread blocks with 512 threads each and an upper bound of 512 blocks per grid for both fine-grain and coarse-grain computations.

Post-processing of data was performed using *Tecplot*®.

## 3.1 Test Case 1: Shock-Vortex Interaction

A popular unsteady benchmark problem representing the acoustics of shock-turbulence interactions is the interaction of an isolated vortex with a normal shock wave and the subsequent formation of acoustic waves. Several experimental and computational studies [70, 71, 72] have focused on the effect of vortex strength and Mach number upstream of the shock on the generation of sound as well as the level of deformation of the vortex and the shock front. The initial conditions consist of a stationary shock in a square domain given by $[-10, 30] \times [-20, 20]$. A relatively large domain is used to ensure that the sound waves do not reach the boundaries during the course of the simulation. The shock is placed at $x = 0$ with

a freestream Mach number ($M_s$) of 1.2, with the flow going from left to right. The left ($x = -10$) boundary is supersonic inflow while zero gradients are enforced at the left ($x = 30$) boundary. The top ($y = 20$) and bottom ($y = -20$) boundaries are set to be periodic. The domain is initialized with $\rho, u, v, p = 1, 1.2, 0, 1/\gamma$ upstream of the shock and post-shock conditions downstream of the shock given by:

$$\begin{aligned} \frac{p_2}{p1} &= \frac{2\gamma M_1^2 - (\gamma - 1)}{\gamma + 1} \\ \frac{\rho_2}{\rho_1} &= \frac{(\gamma + 1)M_1^2}{(\gamma - 1)M_1^2 + 2} \\ M_2 &= \sqrt{\frac{(\gamma - 1)M_1^2 + 2}{2\gamma M_1^2 - (\gamma - 1)}} \end{aligned} \tag{3.1}$$

where the subscript '1' refers to conditions upstream of the shock and the subscript '2' refers to conditions downstream of it. The simulation is started with just the normal shock solution and is run until convergence is obtained. Once the solution is suitably converged, an isentropic vortex is added to the flow at ($x_v = -4,\ y_v = 0$) for which the density and velocity is given by

$$\begin{aligned} \rho &= \left(1 - \frac{1}{2}(\gamma - 1)M_v^2 e^{1-(r/R)^2}\right)^{\frac{1}{\gamma - 1}} \\ \delta u &= -M_v e^{\frac{1}{2}(1-(r/R)^2)}(y - y_v) \\ \delta v &= M_v e^{\frac{1}{2}(1-(r/R)^2)}(x - x_v) \end{aligned} \tag{3.2}$$

where $M_v$ is the vortex strength, $r = \sqrt{(x - x_v)^2 + (y - y_v)^2}$ is the radial distance from the vortex center and $R = 1$ is the vortex radius. Figure 3.1 shows the domain with the initial and boundary conditions.

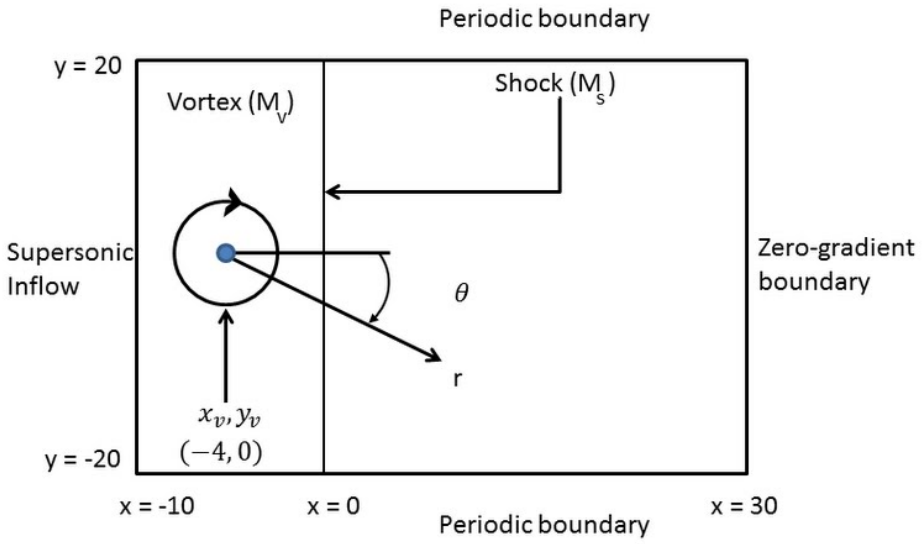
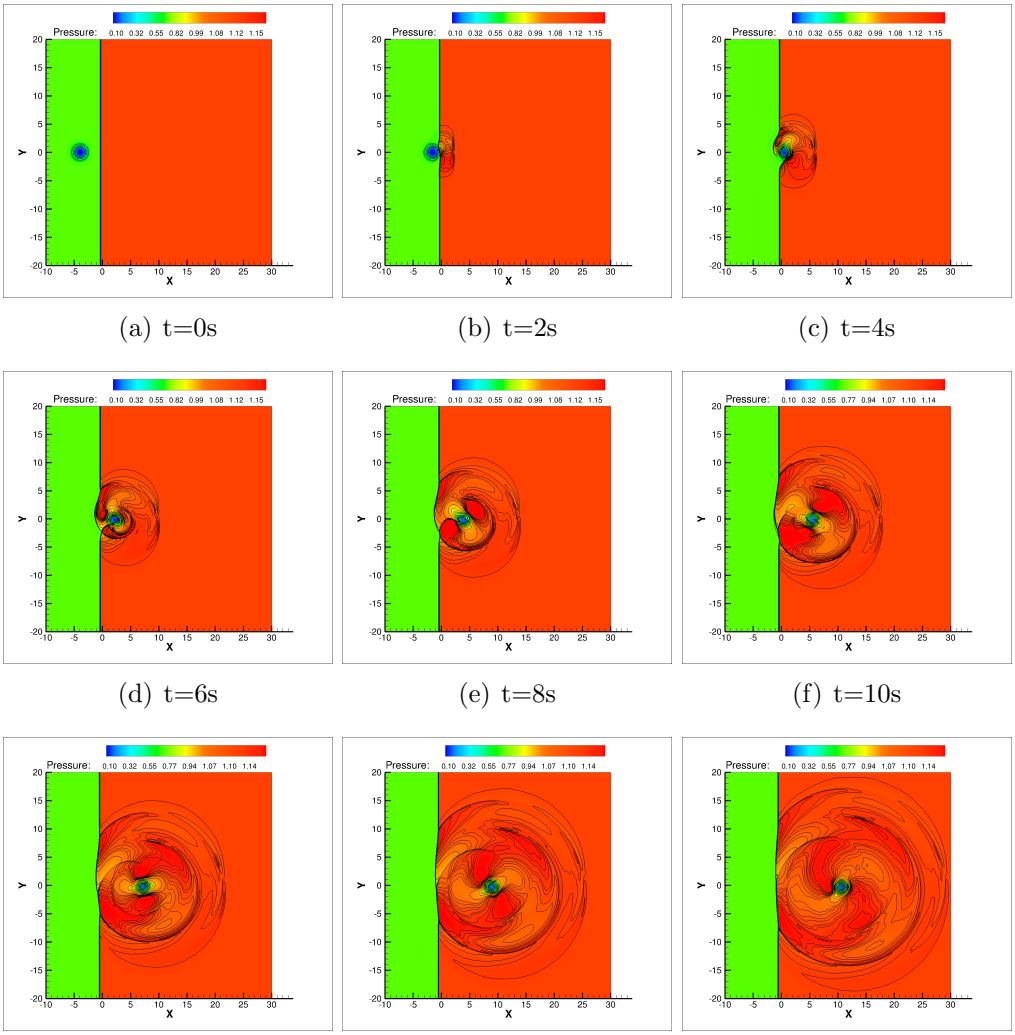



**Figure 3.1:** Schematic diagram of the initial conditions for the shock-vortex interaction

The weak interaction of a vortex ($M_v = 0.25$) with a shock at a Mach number of 1.2 and a Reynolds number of 800 is simulated using the solver. The computational grid used is a uniform 600x600 cartesian grid. Timestepping is done with TVD-RK3 and cell face reconstruction is performed using 5th order WENO.

The ensuing interaction is a multi-stage process in which the primary collision of the vortex with the shock results in the formation of secondary shock structures. Figures 3.2(a)-(h) show the different stages of the shock-vortex interaction.

This test case is used to demonstrate how the GPU performs when only fine-




(a) t=0s (b) t=2s (c) t=4s

(d) t=6s (e) t=8s (f) t=10s

(g) t=12s (h) t=14s (i) t=16s

**Figure 3.2:** Vortex Shock Interaction at M=1.2, Re=800

grain operations are performed. The use of explicit schemes (for both time-stepping and reconstruction) allows the solver to be run entirely in fine-grain mode.

In Fig. 3.3, the predicted variation of sound pressure as a function of radial distance from vortex center at three different instances in the simulation compares well with available experimental measurements.

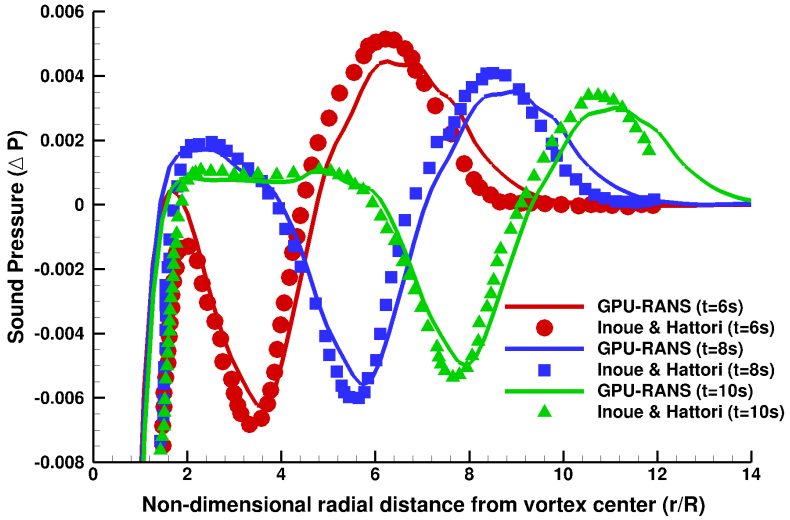



**Figure 3.3:** Variation of sound pressure as a function of distance from the vortex center

As can be seen in Fig. 3.4, the double precision simulation on the GTX480 is found to be 39.69X faster than the double precision simulation in serial mode. In comparison, computations on the GTX640 shows only 5.21X speedup.

Figure 3.5 illustrates the computational expense of the various algorithmic components of GPU-RANS when the solver is run on the GTX640.

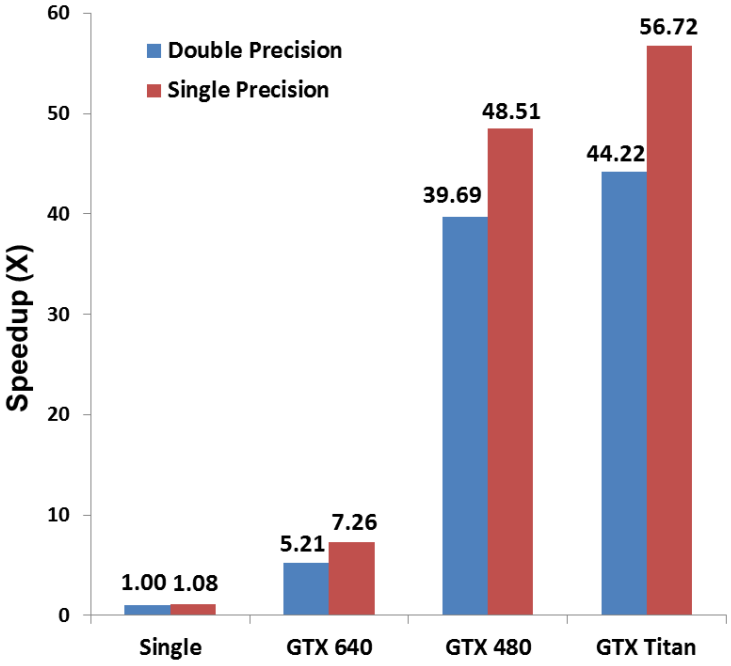



**Figure 3.4:** Compute time comparison on different GPU platforms compared to serial computations

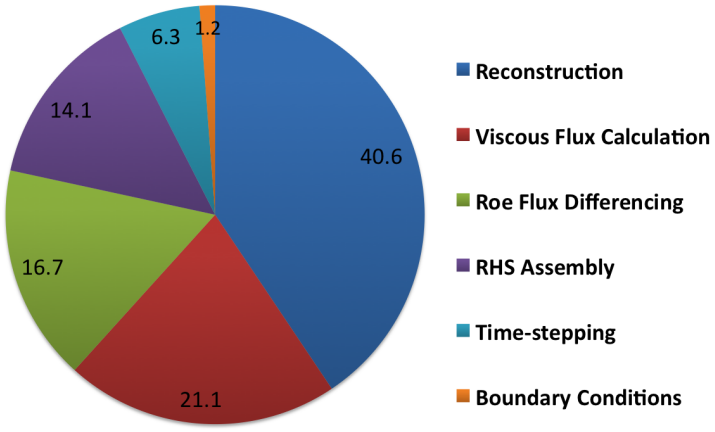



**Figure 3.5:** Profile data on the GTX640 for the shock vortex interaction test case

Among the various fine-grain operations, the WENO5 reconstruction algorithm is clearly seen to be the most expensive.

## 3.2 Test Case 2: Transonic RAE2822 Airfoil

Transonic flow past the RAE2822 airfoil [73] is a popular benchmarking study for CFD solvers. The database associated with this experiment provides a wealth of data for validation purposes such as pressure distribution, boundary layer profiles and wake deficit information. Figure 3.6 shows the experimental setup used in the original study by Cook et al.

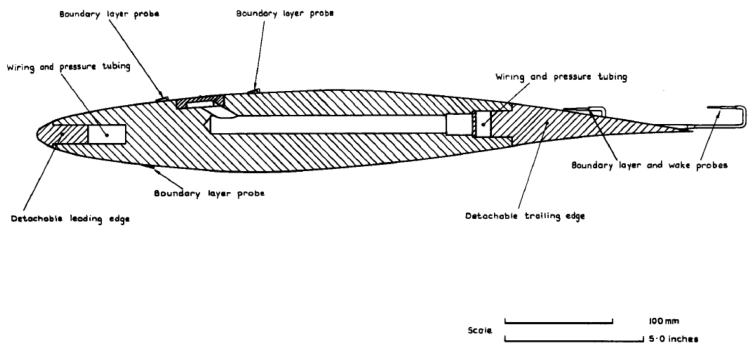



**Figure 3.6:** A cross section of the airfoil used in the experimental study by Cook et al illustrating the various suction ducts and probes used [73]

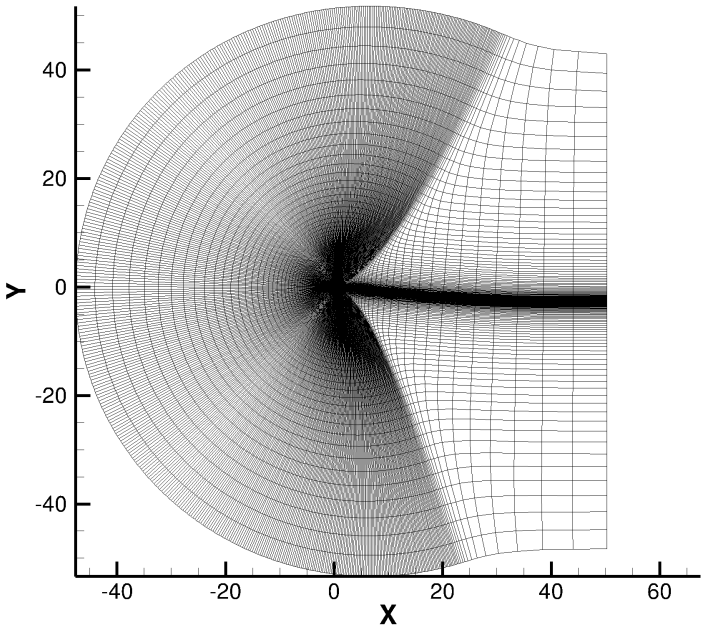
The steady, transonic, turbulent flow around the RAE2822 airfoil is solved to validate the GPU-RANS solver as well as to perform code benchmarking. The domain is discretized by a stretched, C-type mesh with dimensions of 521 X 171 with 402 points on the airfoil surface. The outer boundaries are located 50 chord lengths away. Figure 3.7(a) shows the mesh used in the simulation. Figure 3.7(b) shows a magnified view of the same mesh focusing on the region near the airfoil surface.

The Mach number is 0.725, the Reynolds number is 6.5 million (based on airfoil chord length) and the angle of attack is $2.92^o$. The experimental data was obtained inside a wind tunnel and thus, the freestream conditions for the computations are corrected [74]. The corrected angle of attack and freestream Mach number are $2.51^o$ and 0.731 respectively.

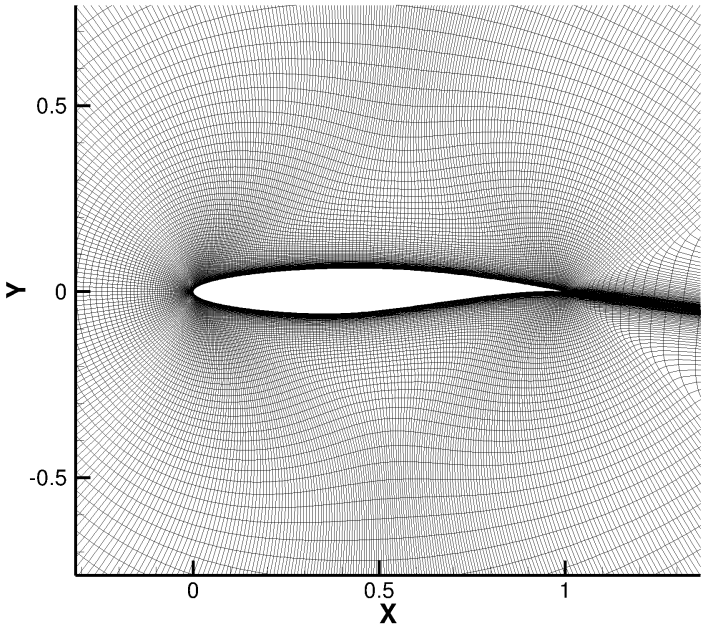
The numerical solution is obtained using the WENO5 scheme for reconstruction and the Euler Implicit scheme (with DADI) is used to march the solution in time to steady state. Unlike in the previous test-case where every participating algorithm was run in fine-grain mode, the use of an implicit time-stepping algorithm introduces a coarse-grain of parallelism into the solver. Furthermore, the implementation of the Spalart-Allmaras turbulence model used in the solver necessitates the solution of an implicit system. This is done using the DDADI (Diagonally Dominant Alternating Direction Implicit) algorithm [58] and is also handled in a coarse-grain manner.

Characteristic-based freestream boundary conditions are enforced on the outer boundaries. No-slip wall boundary conditions are applied on the airfoil surface and wake averaging is used in the wake-cut of the C-type mesh.

Figure 3.8 shows the Mach number contours over the airfoil. The locally supersonic flow on the upper surface and the shock that terminates it are clearly visible. The wake deficit behind the trailing edge is also captured. Figure 3.9 shows the distribution of coefficient of pressure over the airfoil surface for the




(a) Mesh used in the simulation with the RAE2822 airfoil




(b) Magnified view of the region near the airfoil

**Figure 3.7:** The C-mesh used in the RAE2822 airfoil simulation

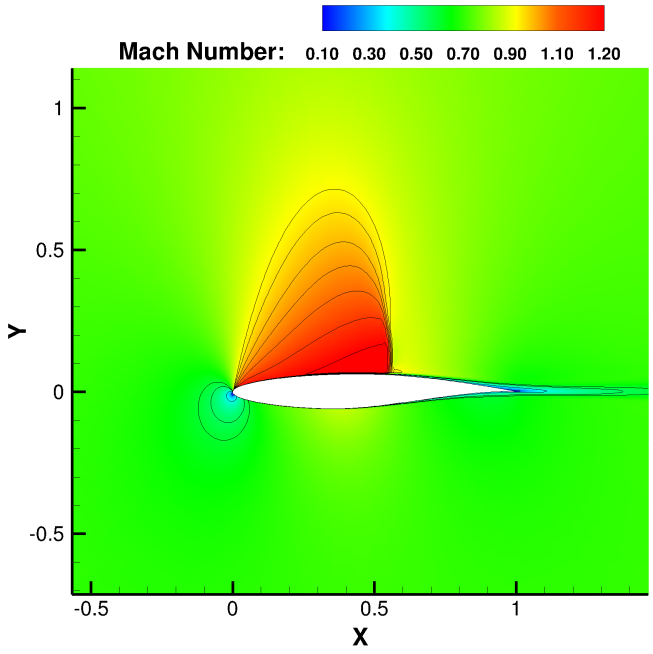



**Figure 3.8:** Mach number contours around a transonic RAE2822 airfoil

computed solution as well as the experimental data [73].

Good agreement is observed between the numerical solution and the experimental data. Figures 3.10(a),(b) show the velocity profiles at two locations: inside the boundary layer on the upper surface at $x/c = 0.319$ and inside the wake at $x/c = 1.025$. Again, the numerical solution is seen to be in good agreement with the experimental data.

Figure 3.11 shows the speedup achieved on various GPU platforms. On the GTX580, double precision computations were seen to be approximately 27 times faster than equivalent single-core computations.

Figure 3.12 shows profile data for the test case illustrating the relative times

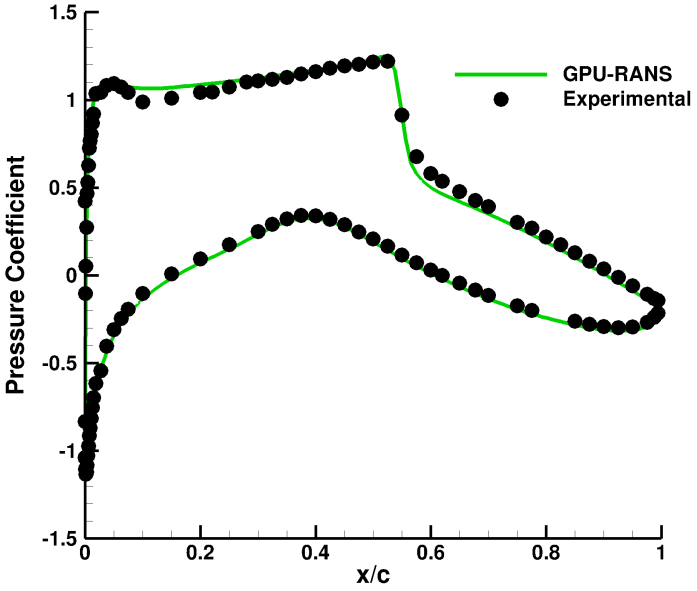
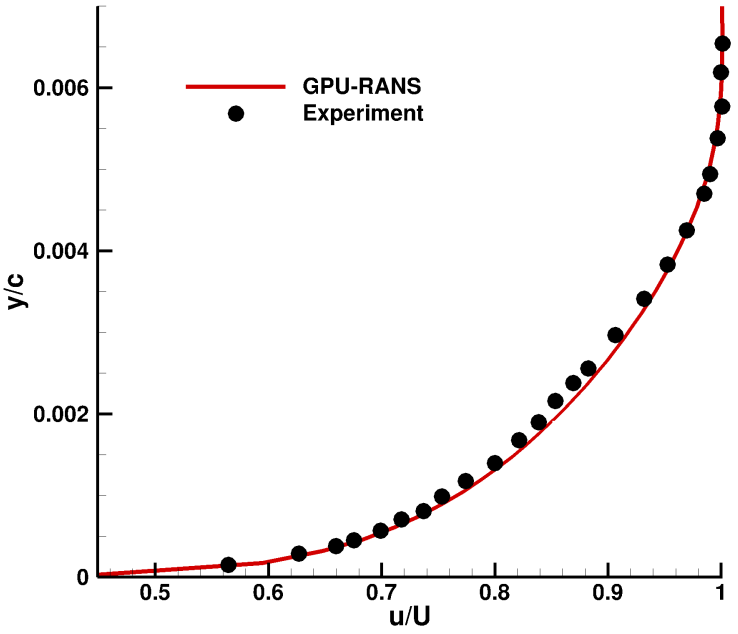
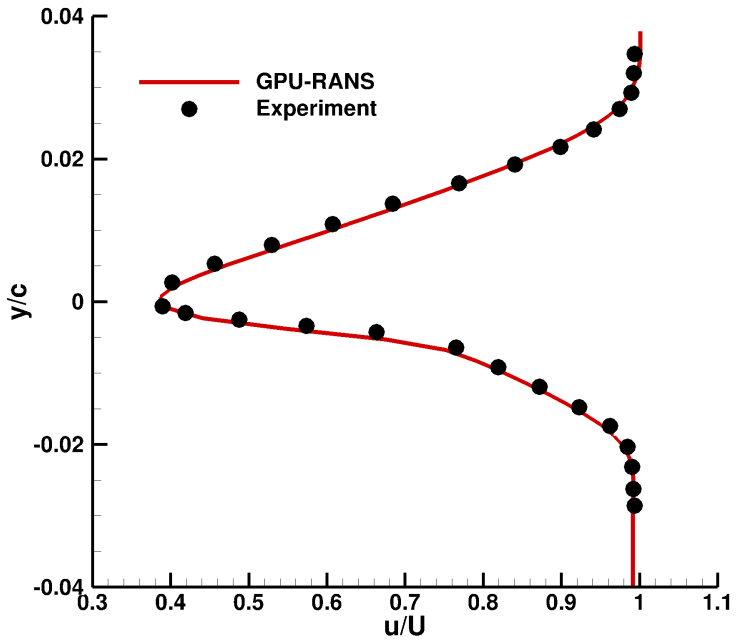



**Figure 3.9:** Pressure coefficient distribution over the RAE2822 airfoil

taken by the various participating algorithms. As expected, the two algorithms that are run in coarse-grain mode - time-stepping and turbulence model - are the two most expensive routines in the solver, accounting for almost 72% of the total compute time.




(a) Boundary layer profile at x/c = 0.319 on the upper surface




(b) Boundary layer profile at x/c = 1.025 in the wake

**Figure 3.10:** Boundary layer and wake deficit profiles

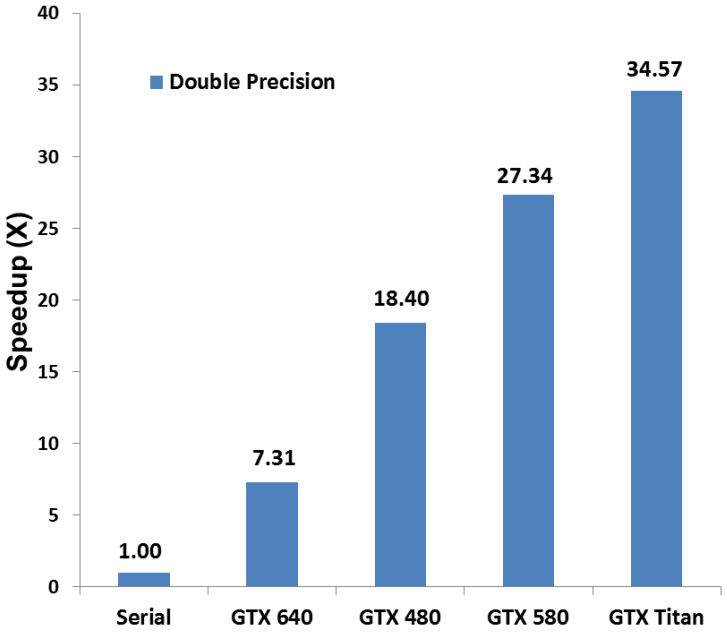



**Figure 3.11:** Compute time comparison on different GPU platforms compared to serial computations

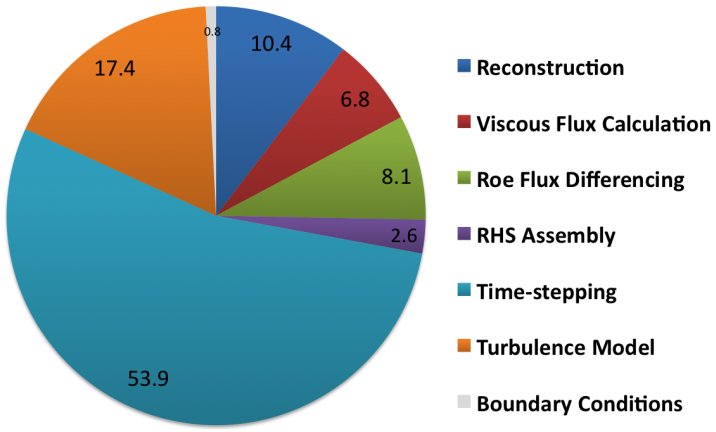
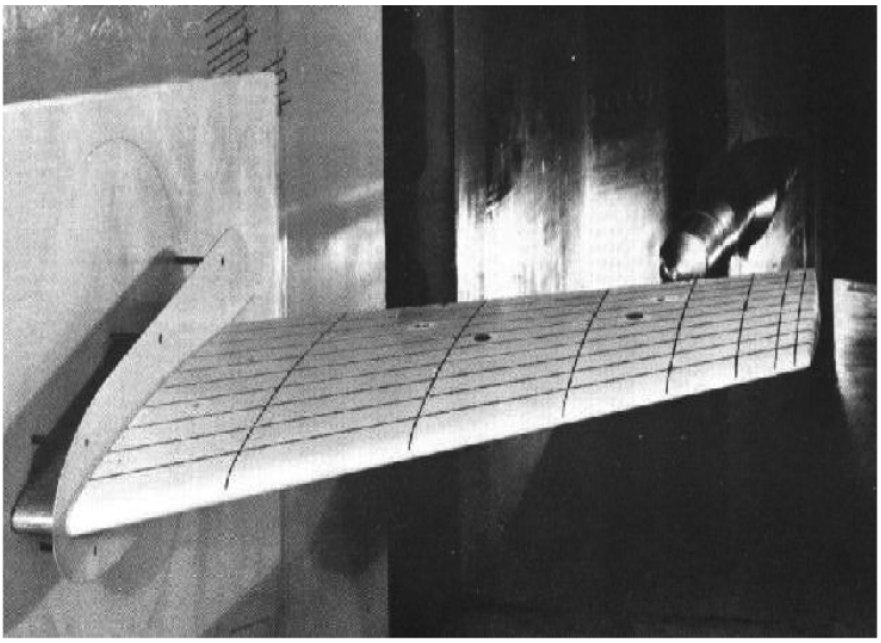



**Figure 3.12:** Profile data on the GTX640 for the RAE2822 airfoil test case

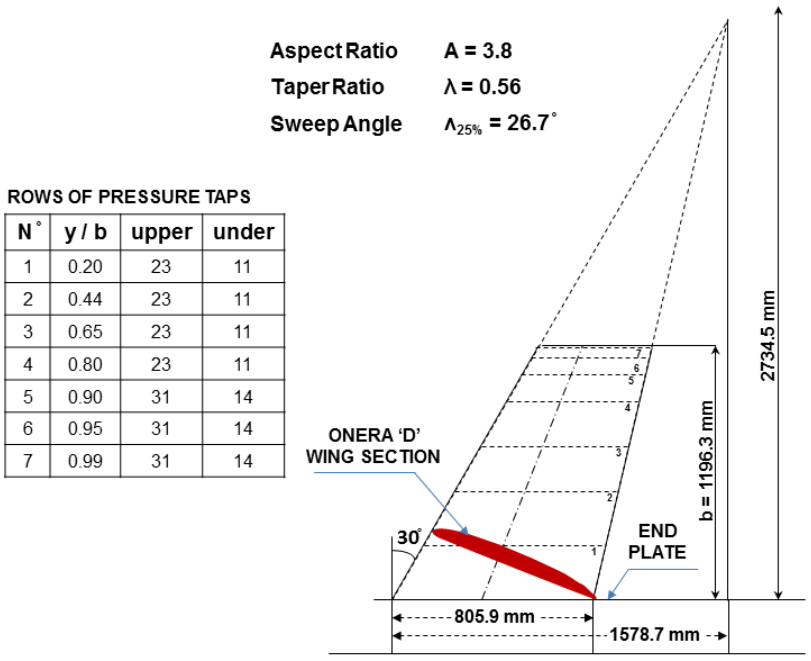
## 3.3 Test Case 3: ONERA M6 Wing

In 1972, the ONERA Aerodynamics Department designed a well instrumented, swept back wing (referred to as the M6-Wing) to be used as an experimental database for basic studies of three-dimensional flows at high Reynolds numbers from low to transonic speeds. Wind tunnel data [75] from this model have provided valuable data for CFD solver validation and for understanding complex flow phenomena such as shock-boundary layer interaction, flow separation and tip-vortex formation. Figure 3.13(a) shows the experimental setup in the wind-tunnel used in the original study and Fig. 3.13(b) shows the placement of the pressure taps on the surface of the blade.

The steady, transonic, flow around an ONERA M6 wing is solved to validate the 3D capability of the GPU-RANS solver as well as to perform code benchmarking for a realistic 3D simulation. The domain is discretized by a single-zone mesh with dimensions of 289x65x49. Figure 3.14 shows the mesh used in the simulation. The outer boundaries are located 50 chord lengths away. Figure 3.14(a) shows the various boundary edges of the domain. The *imin* and *imax* boundaries, shown in green correspond to farfield boundaries. The *jmin* boundary corresponds to the viscous surface of the blade (shown in red) as well as the wake-cut behind the blade (shown in blue). The *jmax* boundary (not shown for the sake of clarity) is also a farfield boundary. The *kmin* boundary is a plane of symmetry and is implemented as an inviscid wall. The *kmax* boundary is a wake-cut boundary beyond the span of the blade. Figure 3.14(b) shows the surface grid of the mesh. Nodes are clustered near the leading edge where large

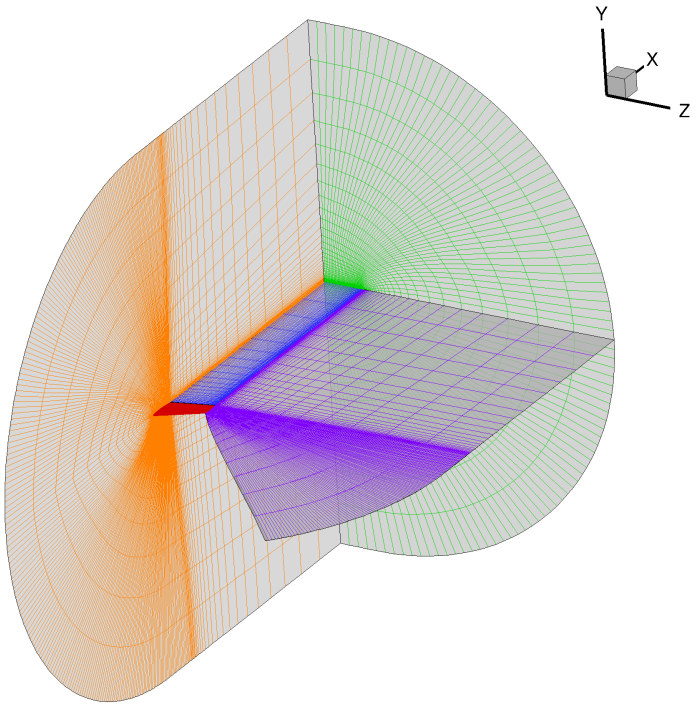


(a) Experimental setup inside the wind-tunnel. Courtesy: AGARD [75]

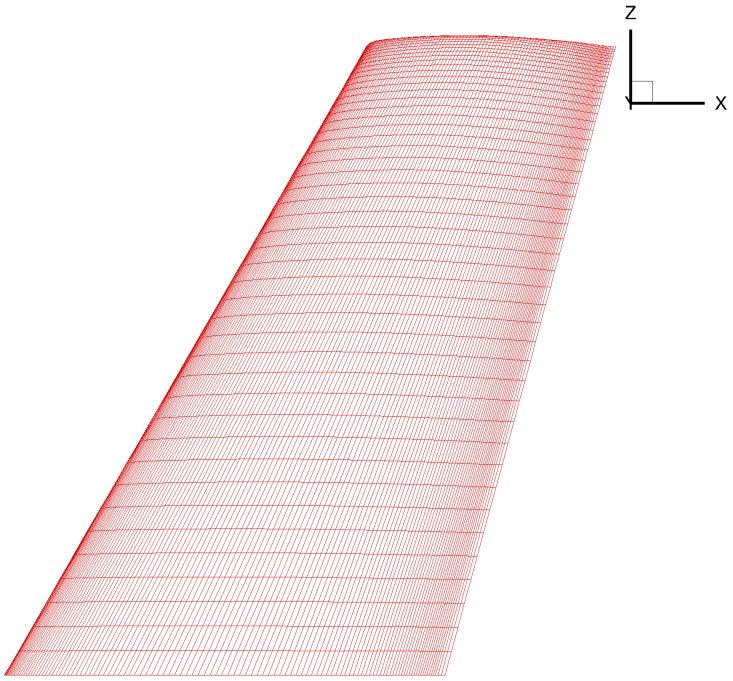



(b) Placement of pressure taps on the blade surface. (Recreated from [75])

**Figure 3.13:** Experimental setup used in the ONERA M6 Wing experiments




(a) Mesh used in the ONERA blade simulation



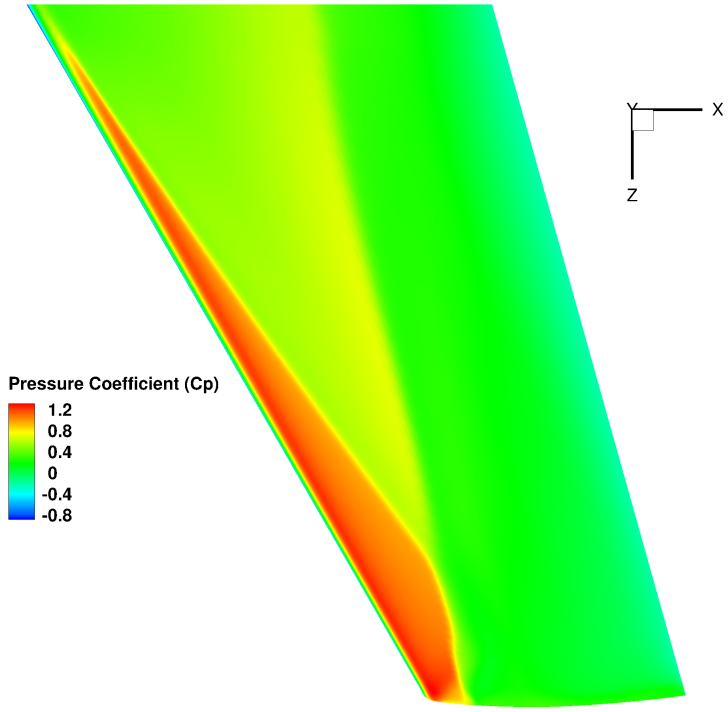

(b) Surface nodes in the ONERA blade mesh

**Figure 3.14:** Single zone mesh system used in the ONERA M6 wing simulation

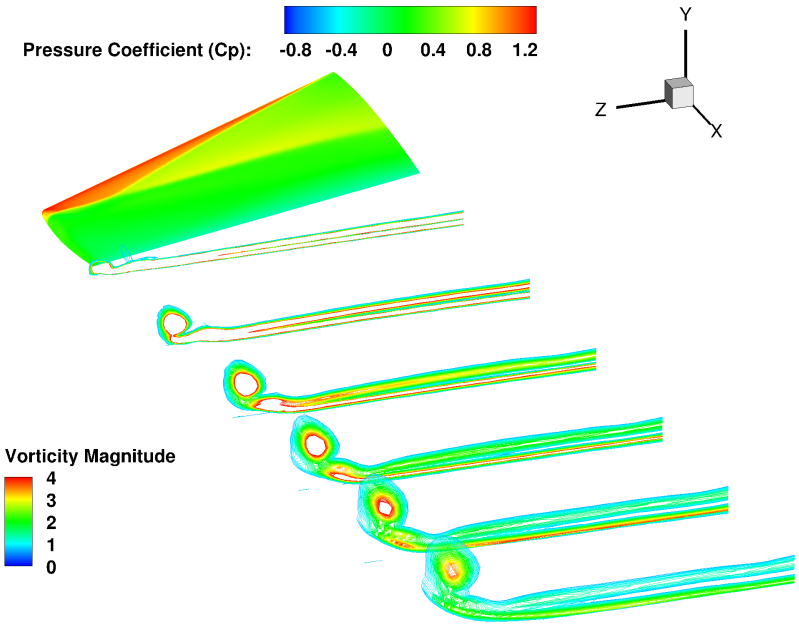
gradients in the flow variables are expected.

The Mach number in the simulation is 0.84, the Reynolds number is 11.72 million (based on mean aerodynamic chord) and the angle of attack is $3.06^o$. The numerical solution is obtained using the WENO5 scheme for reconstruction and the Euler Implicit scheme (with DADI) is used to march the solution in time to steady state. Characteristic-based freestream boundary conditions are enforced on the outer boundaries. No-slip wall boundary conditions are applied on the wing surface and wake averaging is used in the wake-cut regions of the mesh. Figure 3.15(a) shows the variation of pressure coefficient across the surface of the wing. The 'lambda' shock, as seen in experiments, is also captured in the simulation. Figure 3.15(b) shows the formation and convection of the tip vortex downstream of the wing. Figure 3.16(a)-(f) shows the predicted pressure distributions across the chord at six spanwise locations as compared to experimental data.

Figure 3.17 shows the speedup achieved on different GPU plaforms. On the GTX580, double precision computations were seen to be approximately 30 times faster than equivalent single-core computations. Figure 3.18 shows profile data for the test case illustrating the relative times taken by the various participating algorithms. As expected, the two algorithms that are run in coarse-grain mode – time-stepping and the turbulence model – are the most expensive routines in the solver, accounting for almost 58% of the total compute time.

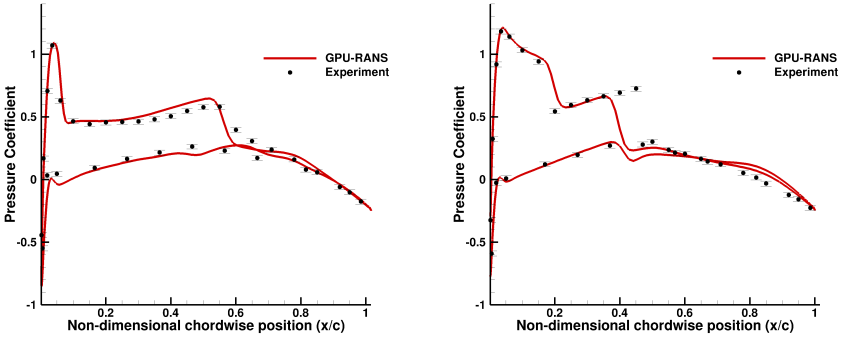



(a) Pressure coefficient contours on an ONERA M6 wing




(b) Tip vortex formation and convection downstream of the ONERA M6 wing

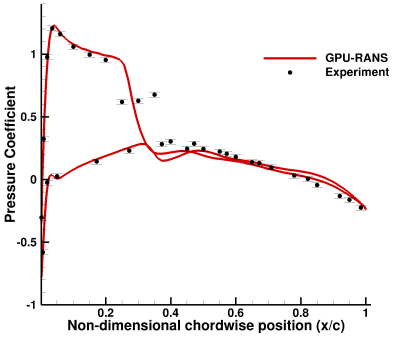
**Figure 3.15:** GPU-RANS predictions of pressure contours and tip-vortex formation in the ONERA M6 wing simulation



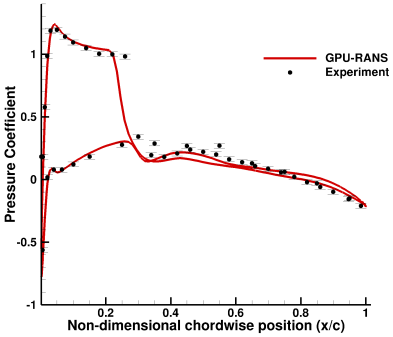

(a) y/L = 0.2



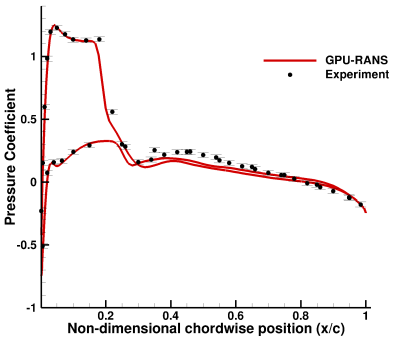

(b) y/L = 0.65



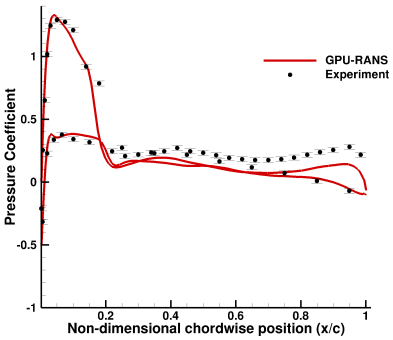

(c) y/L = 0.8




(d) y/L = 0.9




(e) y/L = 0.95




(f) y/L = 0.99

**Figure 3.16:** GPU-RANS predictions of pressure distributions at six spanwise locations compared with experimental data

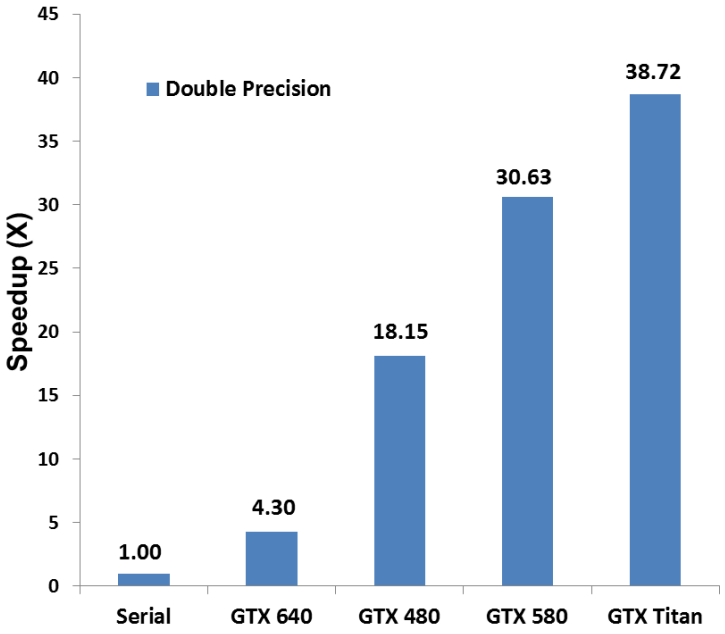



**Figure 3.17:** Compute time comparison on different GPU platforms compared to serial computations

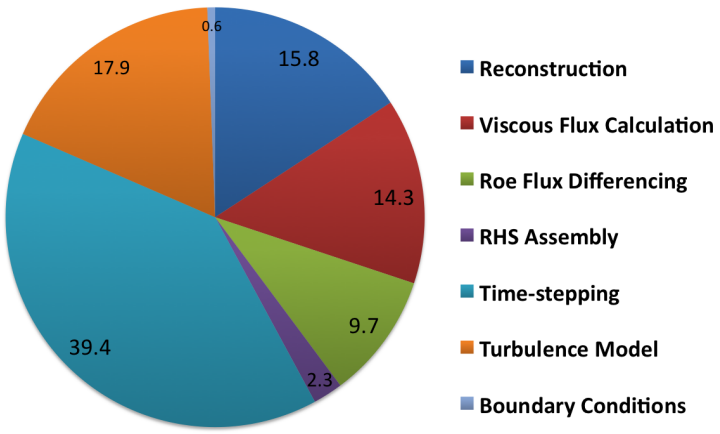



**Figure 3.18:** Profile data on the GTX640 for the ONERA M6 wing case

## 3.4 Test Case 4: Isentropic Vortex Convection

A popular canonical test case for solver verification involves the convection of a 2D isentropic vortex in a uniform inviscid flow-field. A 3D version of this problem is constructed with initial conditions set up to ensure:

1. All gradients in the Z-direction are zero
2. The spatial entropy gradient is zero
3. The velocity, pressure and density fields form an exact solution to the Euler equations

The exact solution to the isentropic vortex convection problem is the pure advection of the vortex at free-stream velocity without any dissipation. In a numerical simulation, as a result of discretization, both dissipative and dispersion errors will be present. Hence, this is a good test case to estimate the numerical diffusion and dispersion characteristics of the algorithms used.

The long-term, inviscid convection of an isentropic vortex is simulated using GPU-RANS. The solution is obtained on a uniform 64x64x64 Cartesian grid in a domain of $[0, 10] \times [0, 10] \times [0, 10]$. Periodic boundary conditions are set at all six boundary surfaces. This is done in order to eliminate the effect of boundary inaccuracies and also to keep the domain size small. Free-stream conditions are given by $\rho, u, v, p = 1, u_\infty, 0, 1/\gamma$. Perturbations are added to the free-stream such that there is no entropy gradient in the flow-field. The perturbations are given

by:

$$\begin{aligned}
\rho &= \left(1 - \frac{(\gamma-1)\beta^2}{8\gamma\pi} e^{1-(r/R)^2}\right)^{\frac{1}{\gamma-1}} \\
p &= \frac{\rho^\gamma}{\gamma} \\
\delta u &= -\frac{\beta}{2\pi} e^{\frac{1}{2}(1-(r/R)^2)}(y - y_v) \\
\delta v &= \frac{\beta}{2\pi} e^{\frac{1}{2}(1-(r/R)^2)}(x - x_v)
\end{aligned} \tag{3.3}$$

Two different reconstruction schemes are tested - 5th order WENO and CR-WENO. Time marching is performed using TVD-RK3. Figure 3.19 shows iso-surfaces of density around the vortex at the start of the simulation. Figures 3.20(a) and 3.20(b) show density contours at one spanwise plane after the vortex has travelled 200 core-radii. Clearly, the higher order, compact reconstruction performs significantly better at preserving the vortex signature. Figure 3.20(c) quantifies this better performance by comparing the variation of absolute error in pressure as a function of time using the various reconstruction schemes.

Figure 3.21 shows that, when CRWENO is used for reconstruction, double precision simulation on the GTX580 is 34X faster than the serial solver. Equivalent computations on the GTX Titan showed a speedup of 46.21X. Compared to the previous 2D test cases, the GPU-RANS solver exhibits higher performance gains in 3D simulations involving coarse-grain computations. The reason for this is that 3D simulations require the creation and inversion of a larger number of linear systems, entailing the use of more GPU threads. The more the number of

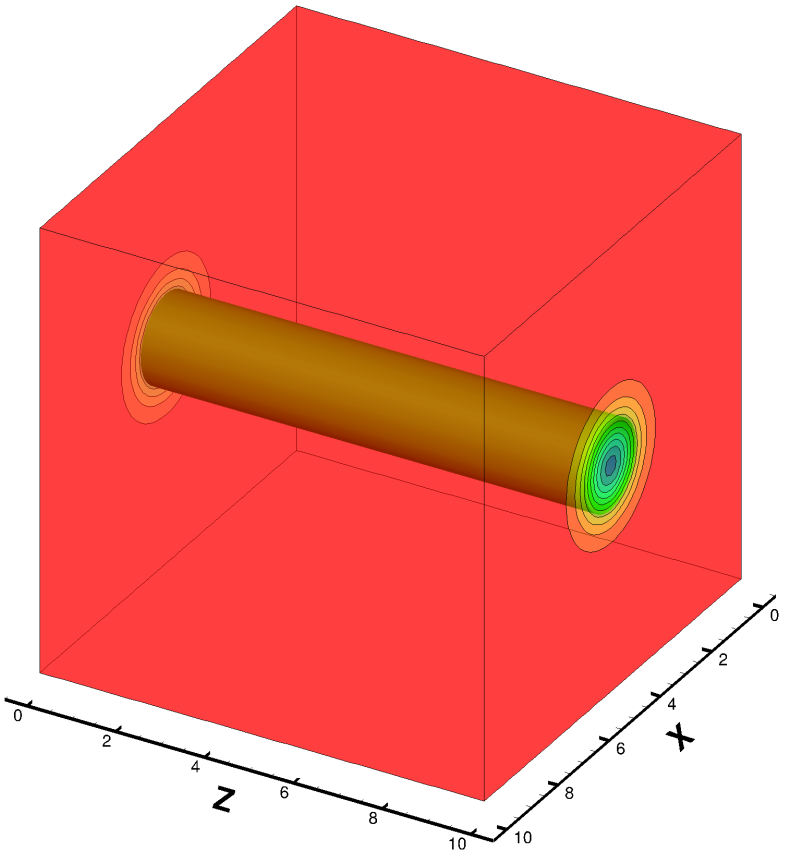
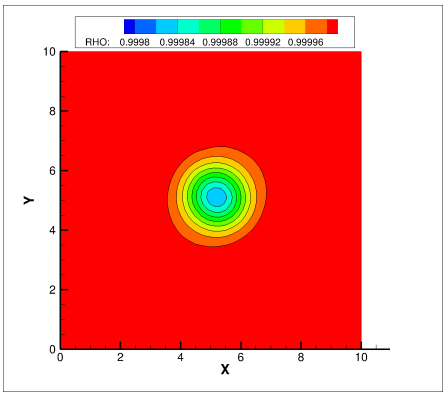
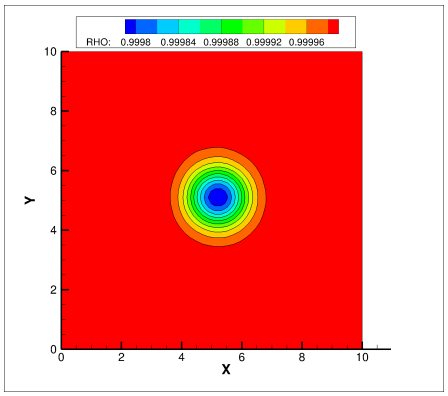


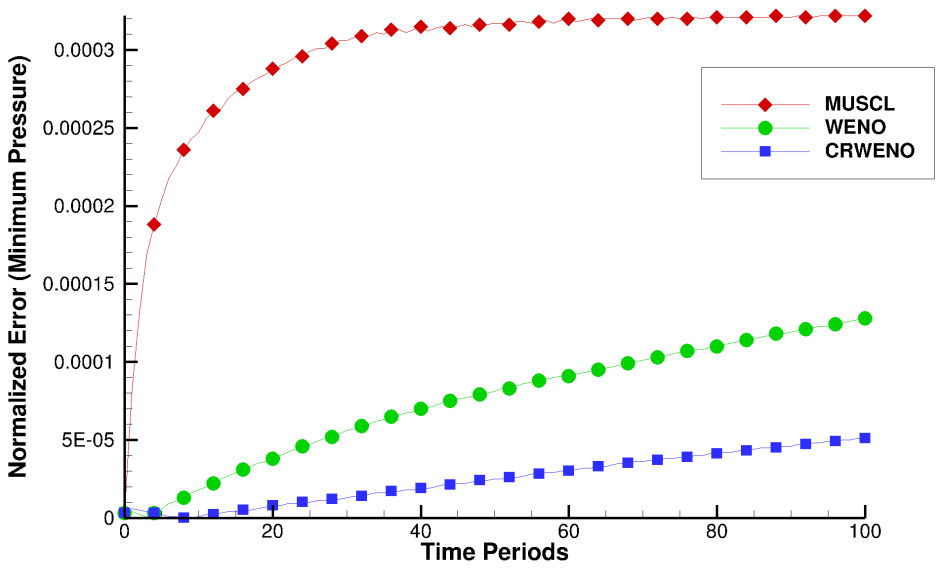

**Figure 3.19:** Isosurfaces of density around a 3D isentropic vortex convecting along the X axis




(a) Computed density contours (WENO5)




(b) Computed density contours (CRWENO)




(c) Variation of normalized error in minimum pressure as a function of time

**Figure 3.20:** GPU-RANS validation and benchmarking with the 3D isentropic vortex

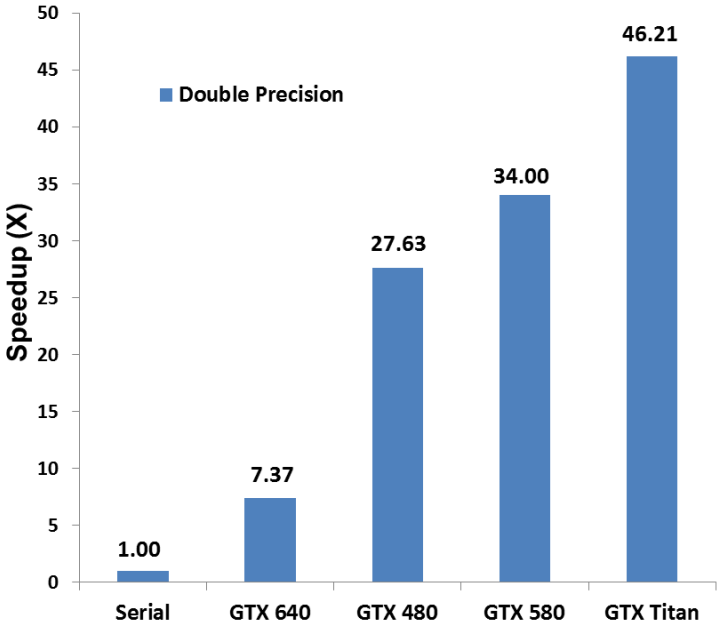



**Figure 3.21:** Compute time comparison on different GPU platforms compared to serial computations

threads, the greater the GPU cores are utilized leading to better latency hiding on the device.

Figure 3.22 shows profile data for the test case illustrating the relative times taken by the various participating algorithms. As expected, the one algorithm that is run in coarse-grain mode -reconstruction - is the most expensive routine in the solver, accounting for almost 69% of the total compute time.

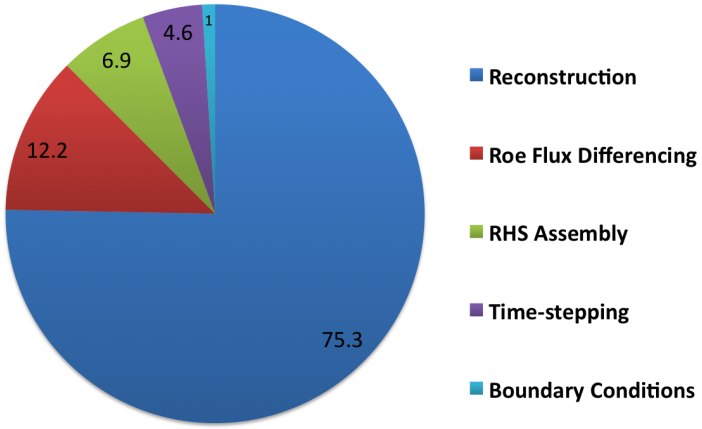
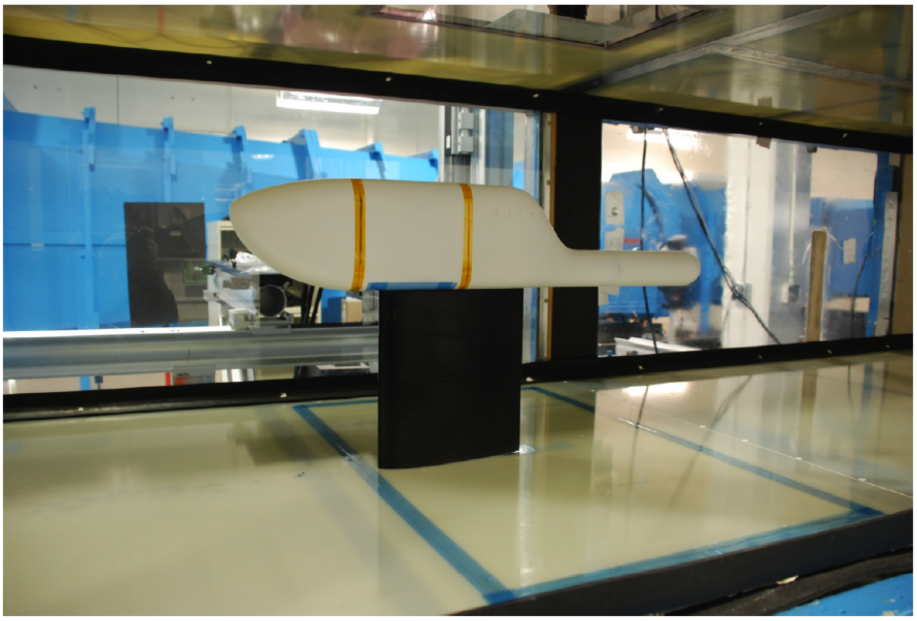



**Figure 3.22:** Profile data on the GTX640 for the isentropic vortex case with implicit reconstruction

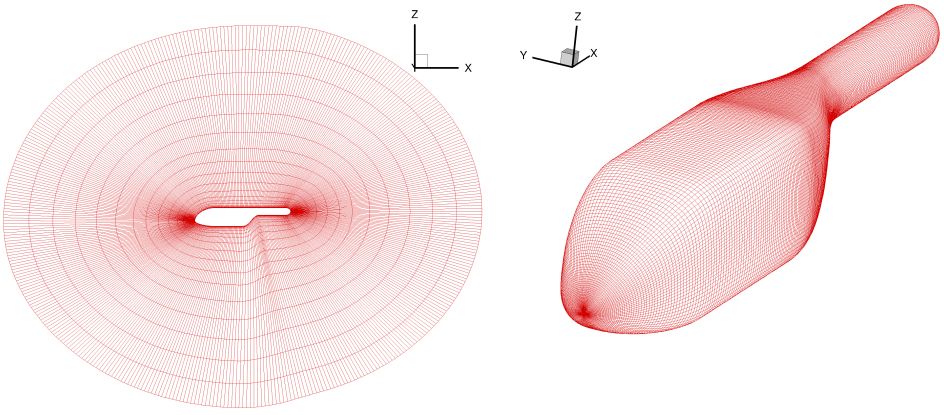
## 3.5 Test Case 5: Isolated Robin-mod 7 fuselage

The ROBIN fuselage was developed at NASA Langley in the 1970s to be a model that was representative of a generic helicopter [76]. A primary motivation of this study was the design of a model that was easily reproducible for computational purposes. Several wind tunnel investigations [77, 78] have made use of this fuselage and it is also popular in the rotorcraft CFD simulation community [69, 79]. In [80], the standard coefficients that describe the original ROBIN fuselage shape, which had a square cross-section, were modified to create a new shape that was more rectangular. In addition, a well defined ramp section and a high tail boom were added. This modified version of the ROBIN fuselage is referred to as the ROBIN-mod7 and is shown in Fig. 3.23(a).

The steady, turbulent flow around the Robin-mod 7 fuselage is solved to validate the GPU-RANS solver as well as to demonstrate the capability of the latest



(a) Experimental setup of the Robin mod-7 fuselage




(b) Azimuthal slice of the grid used in the Robin fuselage simulation

(c) Surface mesh of the grid used in the Robin fuselage simulation

**Figure 3.23:** Experimental setup and mesh used in the Robin mod-7 fuselage

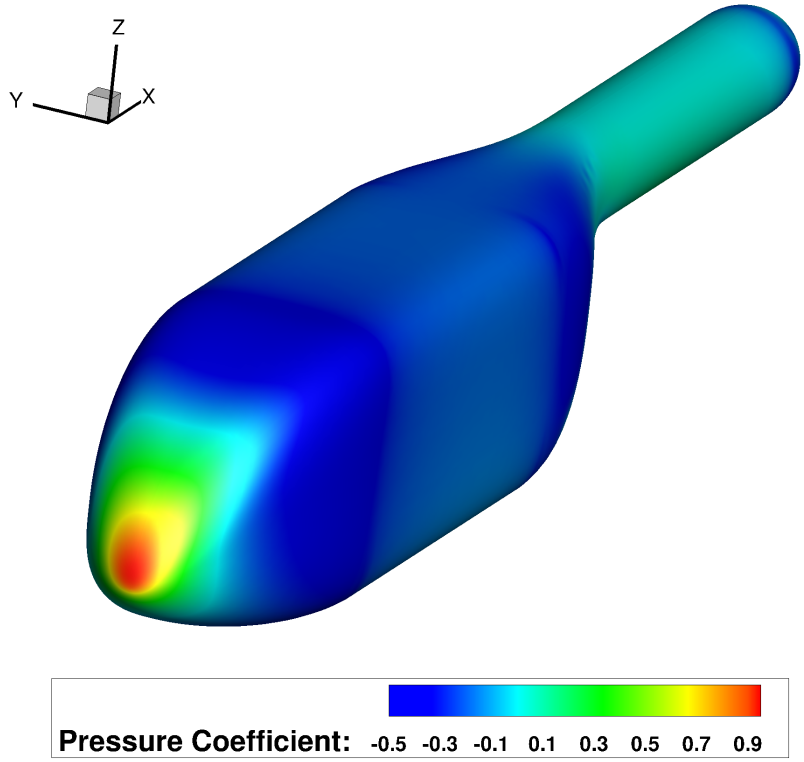
GPU technology to handle large CFD grids. The domain is discretized by a stretched, O-O mesh with dimensions of 321 X 121 X 121. Figure 3.23(b) shows a spanwise slice and Fig. 3.23(c) shows the surface nodes of the mesh. A grid of this size is too large to be run on most gaming GPU platforms. The GTX Titan, however, comes with a VRAM of 6 GB making it a very attractive choice for running relatively large simulations. The outer boundaries are located 15 span lengths away. The freestream Mach number is 0.1, the Reynolds number is 1.6 million (based on fuselage length) and the angle of attack is $0^o$.

Figure 3.24(a) shows the pressure coefficient variation across the fuselage surface. A comparison of predicted variation of pressure coefficient across the upper and lower surfaces along the center-line with experimental data is shown in Fig. 3.24(b). The prediction is shown to be in reasonable agreement with experiment. The peak in pressure just before the ramp and the flat pressure in the separated region on the ramp are well captured.

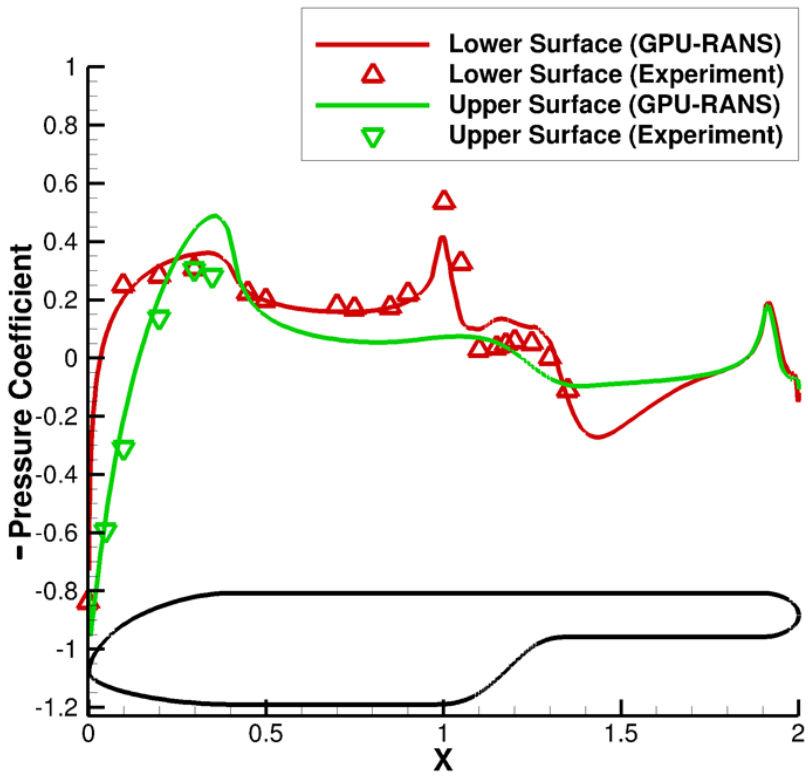
Computations on the GTX Titan were seen to be approximately 36 times faster than equivalent serial computations. A presentation of profile data is omitted since the algorithms used here are identical to those used for the ONERA M6 wing case.

## 3.6 Summary

In this chapter, the GPU-RANS solver was verified and validated. A thorough benchmarking study was performed on both 2D and 3D problems to estimate the




(a) Pressure coefficient contours on fuselage surface






(b) Comparison of pressure coefficient predictions with experimental data

**Figure 3.24:** GPU-RANS validation with the Robin mod-7 fuselage

performance gains that one might expect when algorithms of different granularity were employed in the solution process. It was found that the best performance gains were achieved when the solver was run in a fully fine-grain mode. When coarse-grain computations were introduced for algorithms such as time-stepping, reconstruction or the turbulence modeling, an associated performance penalty was observed. However, the choice of line-parallel techniques for all these algorithms ensured that the performance penalties encountered were not prohibitively large, with performance gains of 30X and above still possible on modern GPU platforms for 3D RANS simulations.

# 4

# Computational Investigation of Brownout Environments

With the GPU-RANS solver validated with an extensive test-suite (described in Chapter 3), the solver can now be used with the other two components of the hybrid methodology – the free vortex method and the particle tracking algorithm – to study single and dual-phase flow environments that are similar to brownout conditions. Detailed analysis of the flow physics is performed and the quality of numerical predictions is tested by comparison with available experimental data. Two different experiments are simulated:

1. Impinging vortex ring

2. Two-bladed, micro-scale rotor

## 4.1 Impinging Vortex Ring

The first test case for validating the hybrid methodology is the experimental setup used by Mulinti et al ([17]). In [17], an impinging wall-jet was forced by modulating the flow at the exit plane of a nozzle to produce a coherent vortex ring that proceeded to interact with the ground plane. This is shown schematically in Fig. 4.1. This vortex ring was created by acoustically forcing the jet using a loud

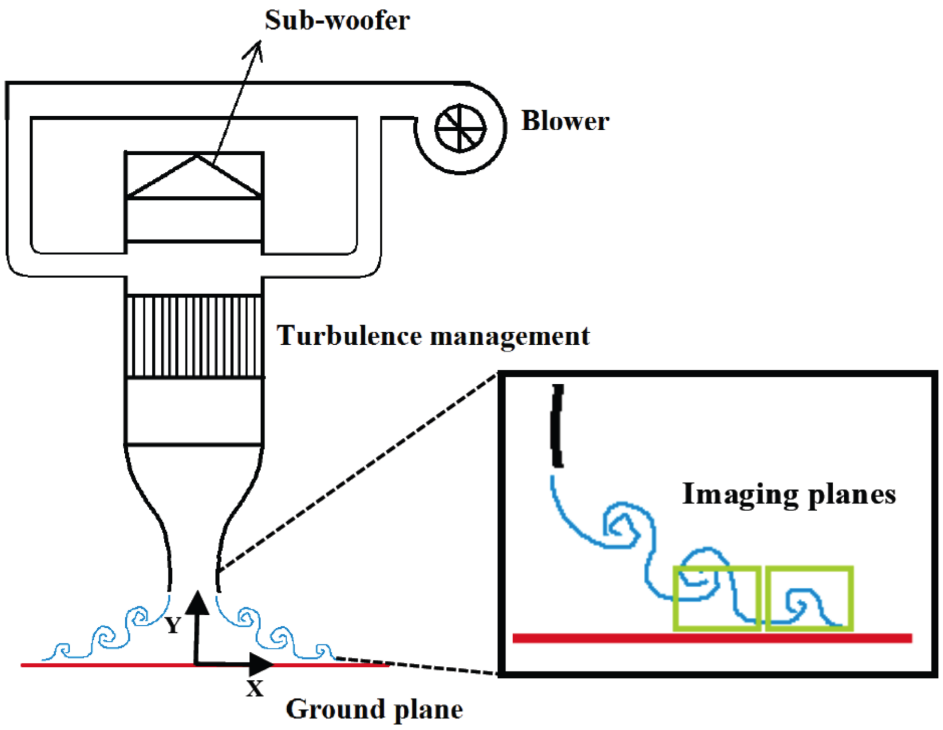



**Figure 4.1:** Experimental setup used by Mulinti et al to study the interaction of an impinging vortex ring with the ground plane [17]

speaker. The flowfield generated consists of a coherent vortex ring superimposed upon an axisymmetric stagnation flow – the two key elements of rotorcraft wakes in ground effect. One key difference between this flow environment and that observed in rotor wakes is that the vortical structures observed are ring-shaped

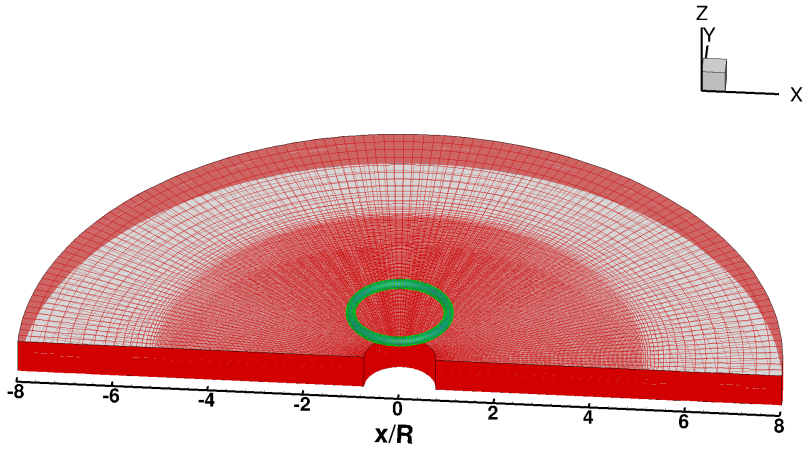
(and axisymmetric) as opposed to helical.

The subwoofer used as the forcing loudspeaker of the jet, shown in Fig. 4.1, has a diameter of 12 and a power output of $2000W$. It uses a pure tone harmonic signal as the forcing input. The test conditions reported in Mulinti's work correspond to a mean jet exit velocity of 4.1 m/s. The amplitude of the forcing was adjusted to yield a peak-to-peak swirl velocity of 12 m/s across the vortex core that was measured to have an initial diameter of approximately 1 cm. The jet was positioned 2 jet nozzle radii (R = 5 cm) above the ground plane.

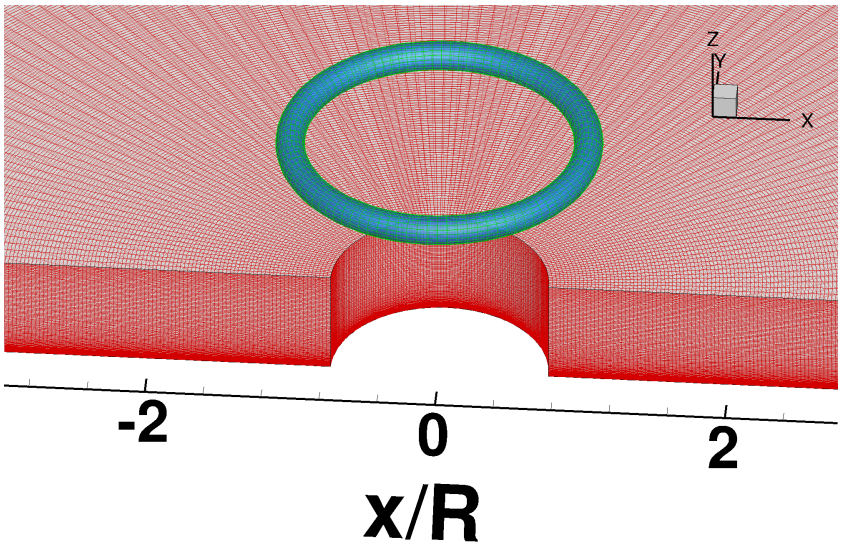
### 4.1.1 Mesh system and Boundary Conditions in GPU-RANS

A single, structured, cylindrical ground mesh, as shown in Fig.4.2, is used for the RANS calculations. Flow periodicity in the azimuthal direction was assumed in order to only need to simulate one half of the computational domain. The dimensions of this mesh in the azimuthal, radial and normal directions are 120 X 180 X 100 points, respectively, and this corresponds to a total of 2.16 million points. In the radial direction, the mesh extends out to 8R while in the ground normal direction, it extends out to 0.8R. This mesh is refined upto 5R to accurately resolve tip vortices all the way to the ground. Beyond this radial location, the mesh is stretched out all the way to the outer boundary at 8R. It is further refined near the ground to resolve the boundary layer.

With the computational grid described in terms of the indices $i, j, k$ where $i$ refers to the azimuthal coordinate, $j$ refers to the radial coordinate and $k$ refers to




(a) Isometric view of the ground mesh used for RANS calculations in the impinging vortex ring simulation




(b) A close-up of the mesh showing the Lagrangian vortex ring initialized at a height of $2R$ above the ground plane

**Figure 4.2:** Mesh system used in the hybrid simulation of the impinging vortex ring

the coordinate in a direction normal to the ground plane, the boundary conditions are applied in the following manner: periodic boundary conditions at $i_{min}$ and $i_{max}$, modified farfield boundary conditions at $j_{min}$, $j_{max}$ and $k_{max}$ and viscous wall conditions at $k_{min}$.

Reconstruction is performed using WENO5 and the second-order BDF scheme is used for time-stepping. A normalized timestep size of 0.02 is used with 6 subiterations per timestep.

On GPU platforms, the GPU-RANS solver employs thread blocks with 512 threads each and an upper bound of 512 thread blocks for both fine and coarse-grain computations. For fine-grain computations, each thread is mapped to a single grid cell. For coarse-grain computations, each thread is mapped to a single coordinate line of grid cells.

### 4.1.2 Lagrangian Wake

The free-wake filaments were initialized in a ring topology with the end of the last filament coinciding with the beginning of the first. The height of the vortex ring was set to twice the nozzle radius corresponding to the height of the nozzle lip above the ground. The temporal discretization in the free-wake solver was chosen to be identical to that used in the GPU-RANS solver ($t^* = 0.02$). The uniform discretization along the filaments was chosen to be $3.6^o$ corresponding to a total of 100 filaments spanning the circumference of the vortex ring.

On GPU platforms, the free-wake solver employs a single thread block with 100 threads with each thread mapped to a single vortex filament for the purpose of induced velocity computation and convection.

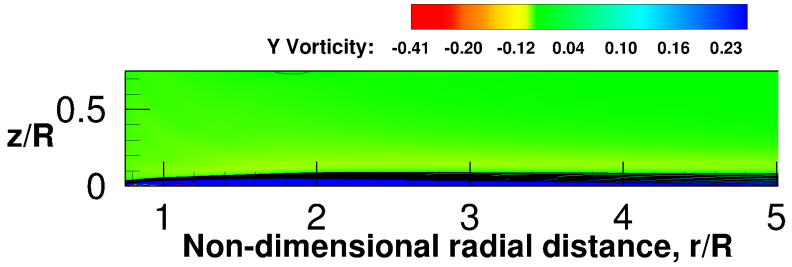
### 4.1.3 Analytical Field

An analytical field of an impinging jet, as described in Chapter 2, that represents the mean flow exiting the nozzle is superimposed upon the vortex ring. The simulation of this field requires the calculation of associated Laguerre polynomials which can be computationally expensive. Fortunately, recurrence formulae for these polynomials exist and are presented in [67]. The use of these recurrence relations greatly reduces the computational burden on the hybrid solver.

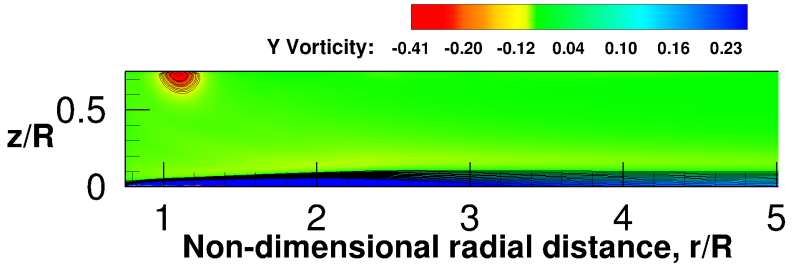
On GPU platforms, the analytical field module employs thread blocks with 512 threads each and an upper bound of 512 thread blocks with each thread corresponding to a single field point where the calculation of the local velocity is required. These field points may be either filaments of the Lagrangian free-wake or boundary cells of the GPU-RANS mesh.

### 4.1.4 Instantaneous Vorticity Contours

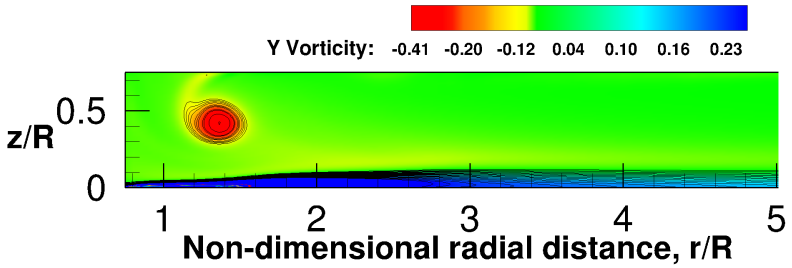
Figures 4.3 and 4.4 show instantaneous vorticity contours near the ground plane at six consecutive time instances. The hybrid methodology is capable of preserving the vortex ring until it begins to interact with the ground plane. This interaction is often accompanied by the creation of opposite-sign vorticity at the ground directly beneath the vortex closest to the ground (eg. 4.4(c)). The boundary layer




(a) $t^* = 64$

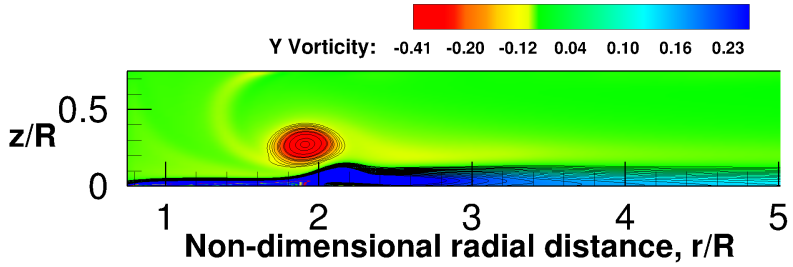



(b) $t^* = 96$

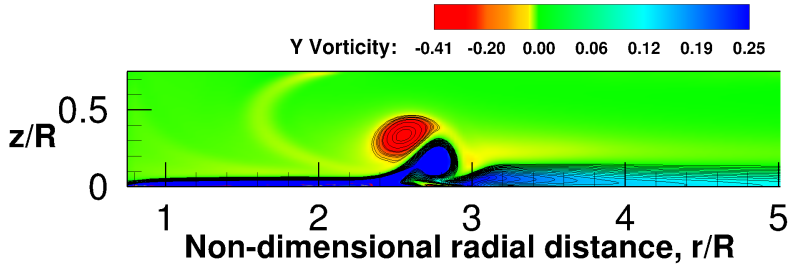



(c) $t^* = 128$

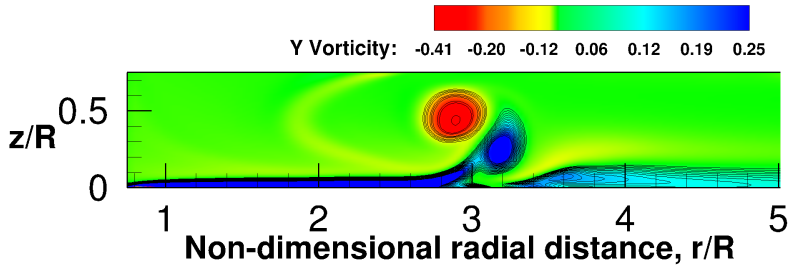
**Figure 4.3:** Instantaneous vorticity contours near the ground plane




(a) $t^* = 160$




(b) $t^* = 192$




(c) $t^* = 224$

**Figure 4.4:** Instantaneous vorticity contours near the ground plane

beneath the vortex is observed to become unstable and separate into a coherent vortical structure. This 'secondary' vortex was observed to have a magnitude that was approximately equal to the strength of the primary vortex directly above it. The formation of a secondary vortex was also observed in experiments by Geiser et al ([81]). Figure 4.5 shows contours of normalized vorticity accompanied by streamlines of velocity as observed in experiments. The radial position at which

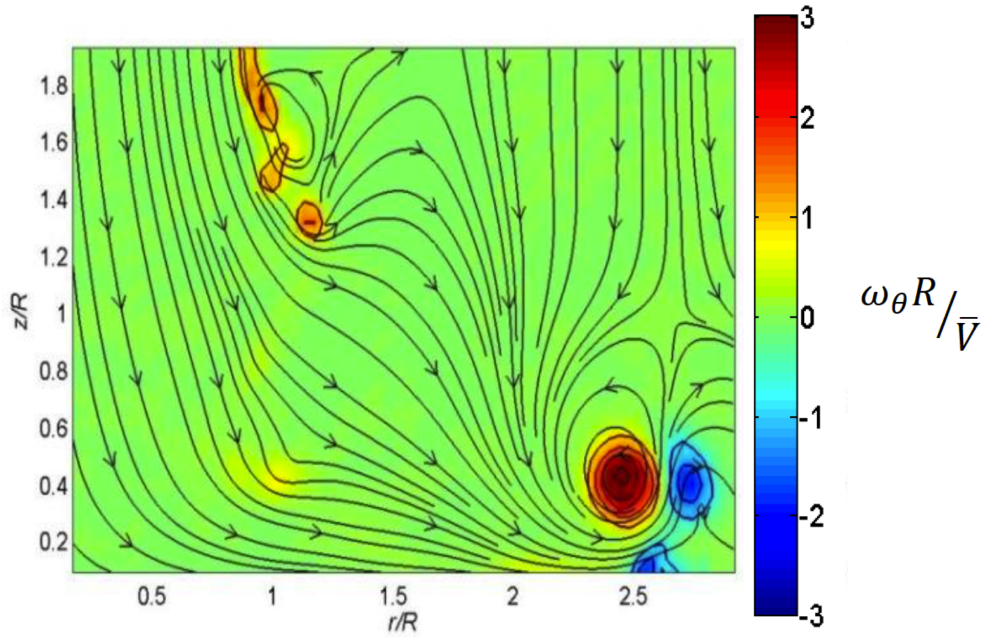



**Figure 4.5:** Contours of normalized vorticity accompanied by velocity streamlines as observed in experiments by Geiser et al [81]

the secondary vortex forms as well as the relative strength of this vortex relative to the vortex ring is found to be in reasonable agreement with computational predictions.

Once instantaneous vorticity contours are computed, the vortex ring trajectory can be approximated. Figure 4.6 illustrates the tip vortex trajectory predicted by the hybrid methodology by superimposing seven representative time instances in

the simulation. The fourth time instance ($t^* = 96$) corresponds to the temporal

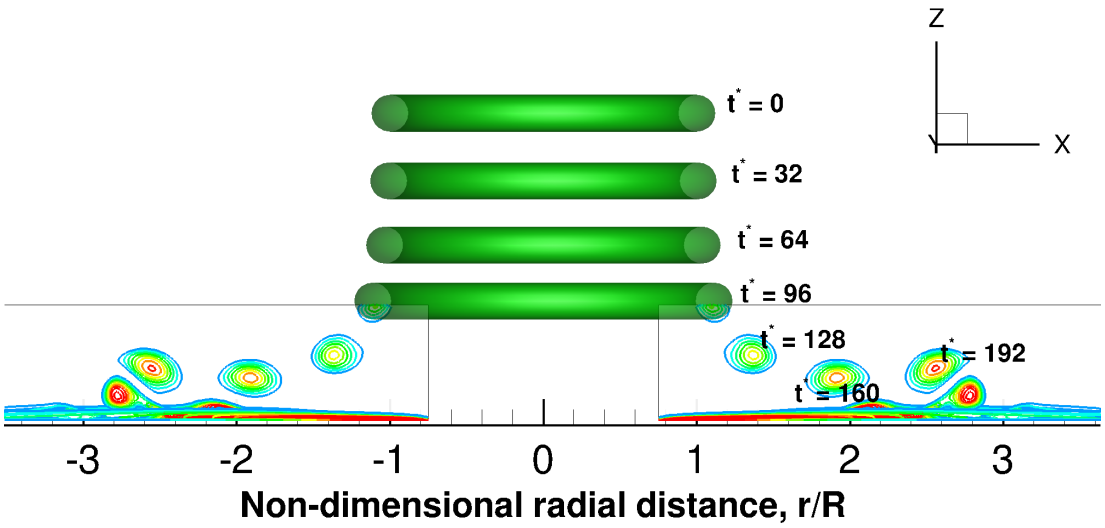
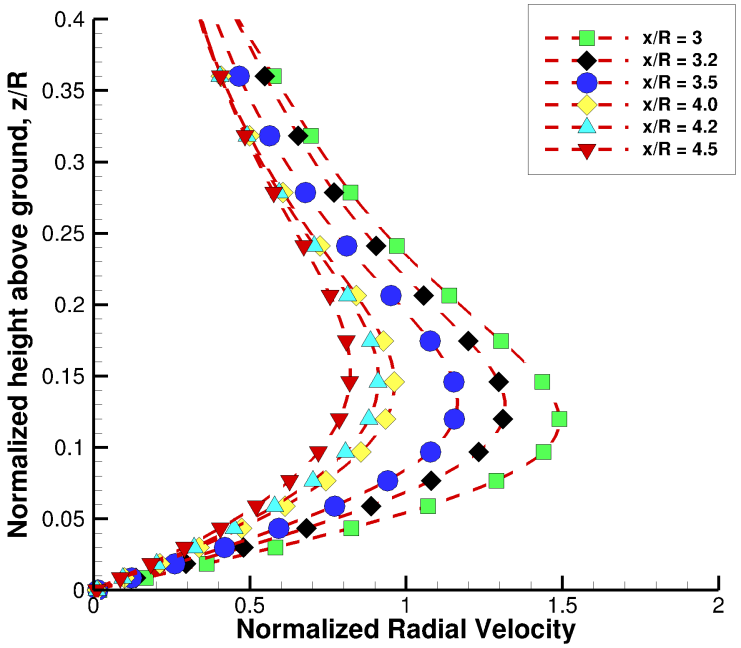



**Figure 4.6:** Hybrid methodology prediction of vortex ring trajectory

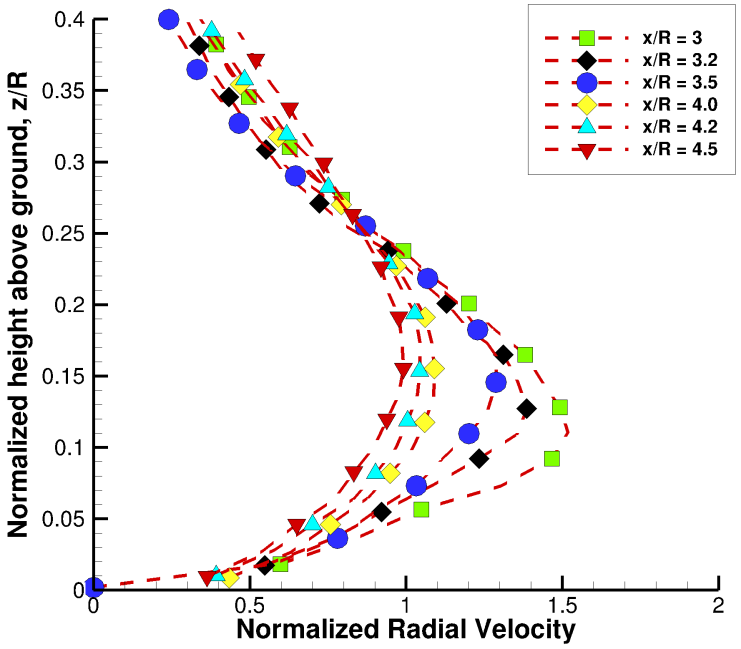
window where the vortex ring transitions from the domain of the Lagrangian free-wake model into the GPU-RANS mesh.

### 4.1.5 Time-averaged velocity profiles

A quantitative validation of the simulated flowfield near the ground plane can be obtained by comparing the time-averaged velocity profiles near the ground with experimental data. Figure 4.7 shows the comparison between the experimental and the computed time-averaged radial velocity profiles at six different radial locations (3.0R, 3.2R, 3.5R, 4.0R, 4.2R and 4.5R) for the jet height of 2R above ground. The computational prediction of peak wall jet velocities as well as the shape of the velocity profiles at these six radial locations is in reasonable agreement with experiment.




(a) Computational prediction of time-averaged radial velocity profiles




(b) Experimental measurement of time-averaged radial velocity profiles

**Figure 4.7:** Experimental and computational measurements of time-averaged radial velocity profiles near the ground plane

## 4.2 Two-bladed Micro-scale Rotor

The second test case used to validate the hybrid methodology is the hovering micro-rotor setup used in [13, 16]. The flow visualization results from Lee et al. [13] are used to validate the single-phase predictions and the results from Sydney et al. [16] are used to validate the dual-phase predictions in the current simulation. The blades are rectangular and untwisted, with a radius of 86 mm and a chord equal to 19mm resulting in an aspect ratio of 4.39. The resulting rotor solidity is 0.145. The untwisted rectangular blades use a 3.3% curvature circular arc airfoil with a thickness of 3.7%. Both the leading and trailing edges are designed to be blunt. The rotor was operated at rotational frequency of 50 Hz which corresponded to a tip speed of 27 m/s, a tip Reynolds number of 32400, a root Reynolds number of 6480 and a tip Mach number of 0.08. The experiments were conducted at rotor heights varying from 0.25R to 1.5R. In the current study, the focus will only be on the 1R rotor height case.

### 4.2.1 Mesh System and Boundary Conditions in GPU-RANS

A single, structured, cylindrical ground mesh, as shown in Fig.4.8(a), is used for the RANS calculations. Flow periodicity in the azimuthal direction was assumed to simulate only one half of the computational domain. The dimensions of this mesh in the azimuthal, radial and normal directions are 60 X 180 X 120 points, respectively, and this corresponds to a total of 1.3 million points. In the radial direction, the mesh extends out to 5R while in the ground normal direction, it extends out to 0.5R. This mesh is refined upto 3R to accurately resolve tip vor-

tices all the way to the ground. Beyond this radial location, the mesh is stretched out all the way to the outer boundary at 5R. It is further refined near the ground to resolve the boundary layer as can be seen in a single azimuthal plane of the mesh (Fig.4.8(b)).

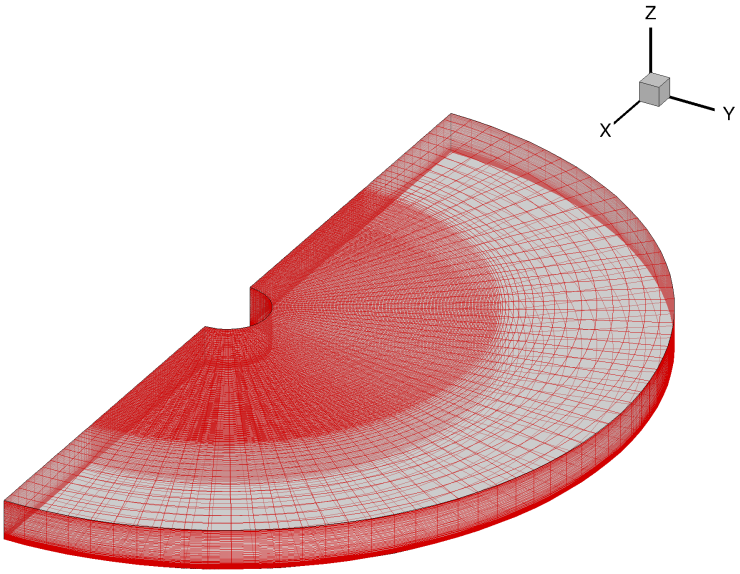
With the computational grid described in terms of the indices $i, j, k$ where $i$ refers to the azimuthal coordinate, $j$ refers to the radial coordinate and $k$ refers to the coordinate in a direction normal to the ground plane, the boundary conditions are applied in the following manner: periodic boundary conditions at $i_{min}$ and $i_{max}$, modified farfield boundary conditions at $j_{min}$, $j_{max}$ and $k_{max}$ and viscous wall conditions at $k_{min}$.

Reconstruction is performed using WENO5 and the second-order BDF scheme is used for time-stepping. A timestep size of 0.1 degree is used with 5 subiterations per timestep to remove factorization errors and recover time-accuracy. Time-stepping is performed using the BDF2 algorithm and reconstruction is performed using WENO5.

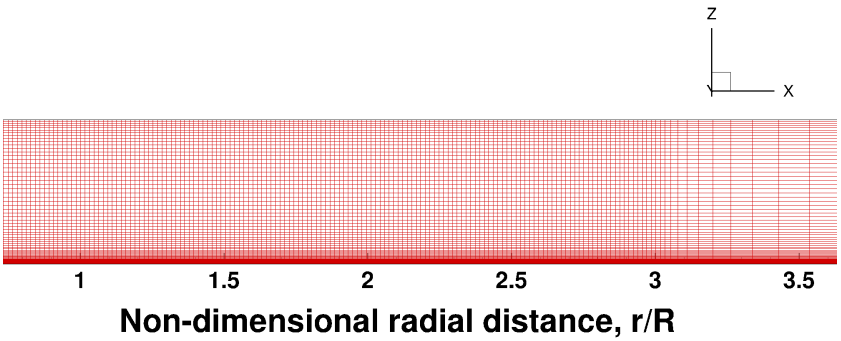
On GPU platforms, the GPU-RANS solver employs thread blocks with 512 threads each and an upper bound of 512 thread blocks.

### 4.2.2 Lagrangian Wake

The temporal discretization in the free-wake solver was chosen to be $1^o$. This timestep size allowed for the free-wake solver to be run once every 10 GPU-




(a) Isometric view of the ground mesh used for RANS calculations in the simulation of the two-bladed, micro-scale rotor




(b) Side view of the ground mesh showing refinement to resolve both tip vortices and the boundary layer

**Figure 4.8:** Mesh system used in the hybrid simulation of the micro-rotor

RANS steps. The uniform discretization along the filaments was chosen to be $5^o$ and a total of four wake-turns were used in the Lagragian free-wake model. The initial core radius of the vortex filaments was set to 5% of chord. Airloads from a converged linear aerodynamics simulation were used as input to the solver and the vortex strengths were set to be equal to the derivative of the bound circulation at the spanwise location corresponding to the maximum sectional thrust.

On GPU platforms, the free-wake solver employs thread blocks with 512 threads each and a total of 2 thread blocks.

### 4.2.3 Sediment Tracking Algorithm

For dual-phase simulations, a circular sediment bed of radius $3R$ is placed directly beneath the rotor. The particles are convected using a second order Runge-Kutta time-stepping scheme with a time-step size of 10 degrees of azimuth. This timestep size allowed for the sediment tracking algorithm to be run once every 100 GPU-RANS steps.

On GPU platforms, the sediment tracking algorithm employs thread blocks with 512 threads each and an upper bound of 512 thread blocks.

### 4.2.4 Performance Prediction

The hybrid methodology uses prescribed airloads to compute tip-vortex strengths of the filaments in the FVM solver which in turn induce a velocity field on the boundaries of the RANS mesh. The quality of the predictions near the ground

plane is therefore strongly dependent on the accuracy of these airloads. The mean thrust distribution across the blade span is computed using linearized aerodynamics with airfoil tables generated by 2D simulations of the micro-rotor blade in [82]. Figure 4.9 shows the predicted variation of lift-coefficient as a function of angle of attack at a Reynolds number corresponding to 60% span of the micro-rotor blade.

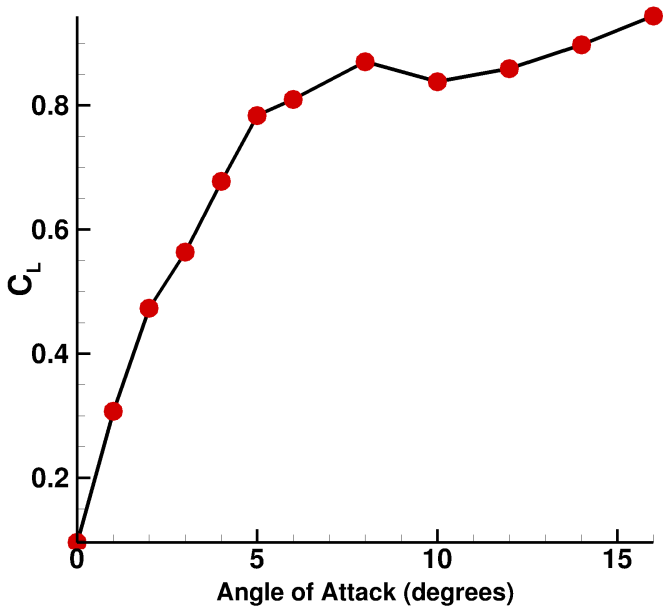



**Figure 4.9:** Variation of lift coefficient of the micro-rotor airfoil with angle of attack

Figure 4.10 shows the thrust distribution predicted by the linear aerodynamics model as a function of spanwise location for the micro-scale rotor at a collective setting of $12^o$. This predicted distribution compares reasonably well with predictions made by a RANS simulation [24]. Integrating this distribution across both rotor blades yields a thrust coefficient of 0.0162. This value is approximately 5% higher than experimental observation.

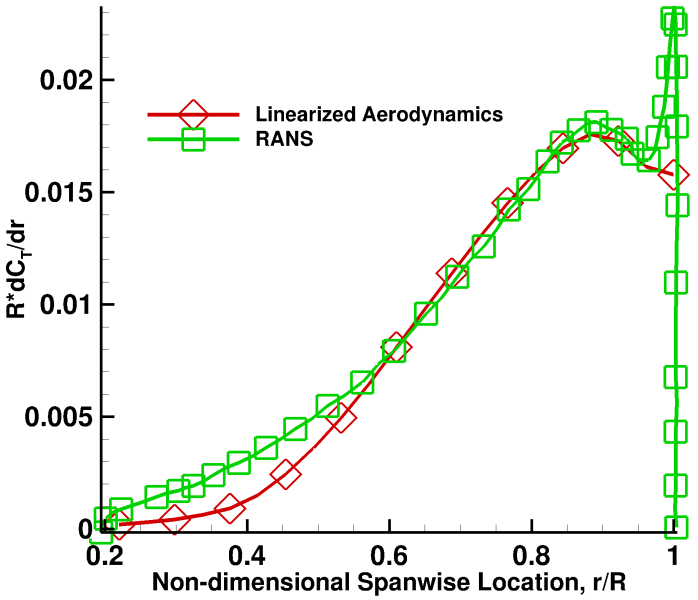
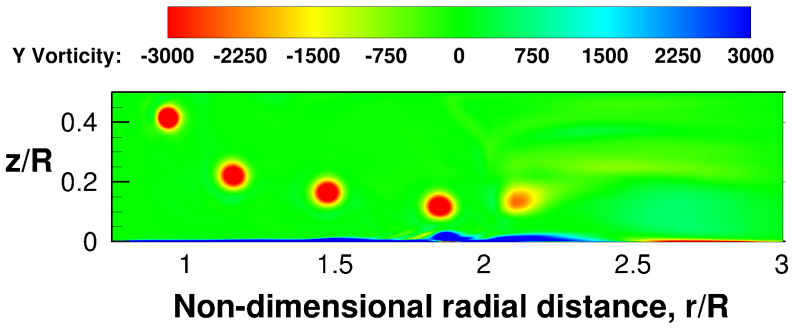



**Figure 4.10:** Spanwise thrust distribution for micro-scale single rotor, at a collective setting of $12^o$

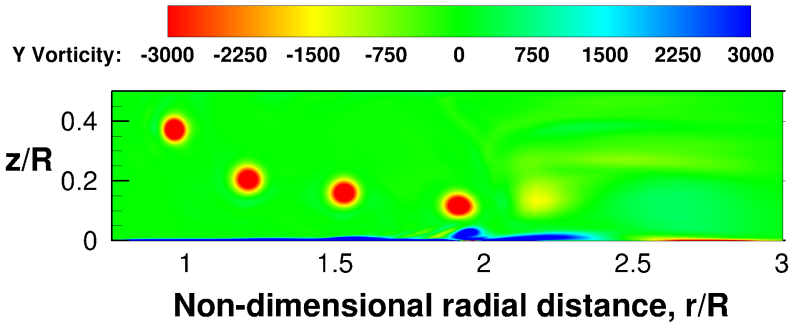
### 4.2.5 Instantaneous Vorticity Contours

Figures 4.11 and 4.12 show instantaneous vorticity contours near the ground plane at six different rotor phase angles.

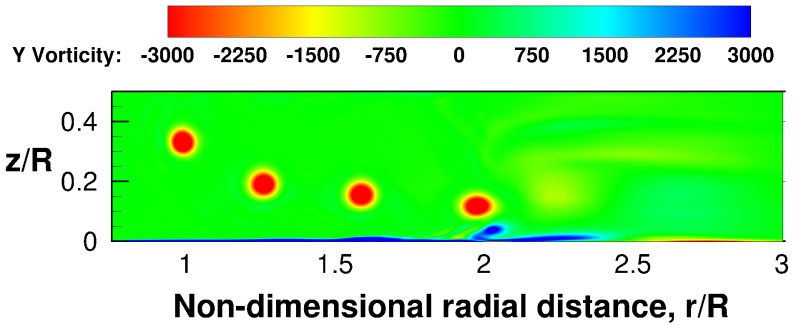
The hybrid methodology is capable of preserving vortical structures across several blade passages until they start to interact with the ground plane. This interaction is often accompanied by the creation of opposite-sign vorticity at the ground directly beneath the vortex closest to the ground (eg. 4.11(c)). The vortex closest to the ground energizes the boundary layer. However, the suction pressure associated with the vortex leads to an adverse pressure gradient eventually causing the boundary layer to become unstable and separate into a coherent vortical structure. This 'secondary' vortex was observed to have a magnitude that




(a) $\Psi = 0^o$

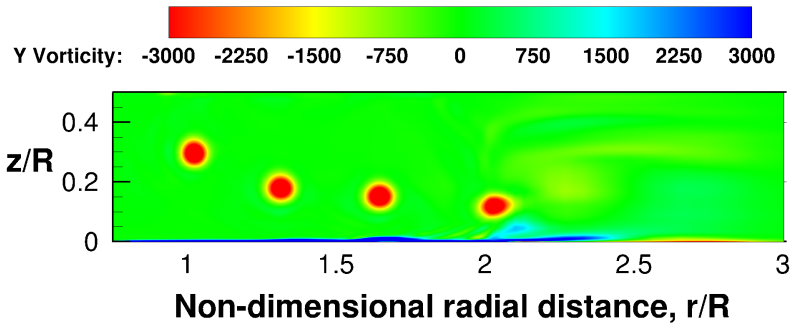



(b) $\Psi = 30^o$

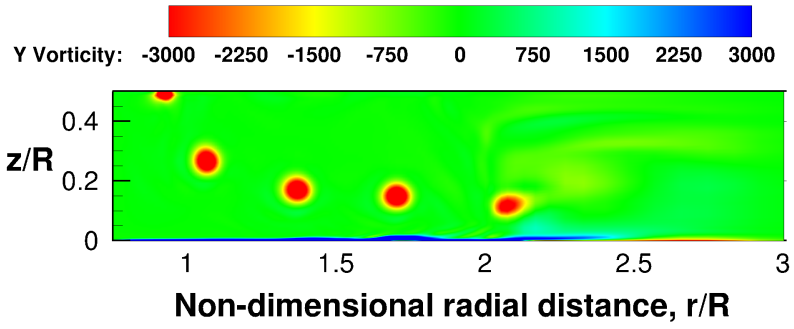



(c) $\Psi = 60^o$

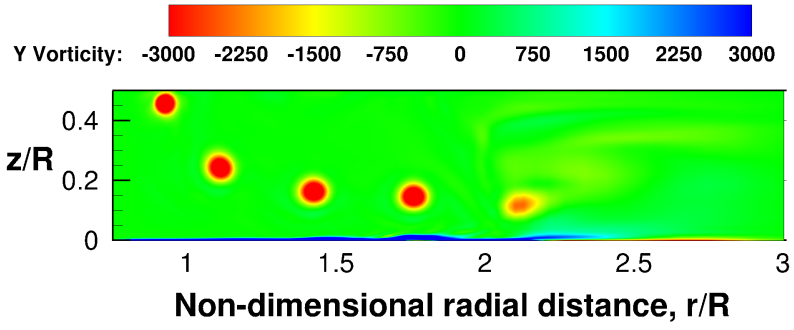
**Figure 4.11:** Instantaneous vorticity contours near the ground plane




(a) $\Psi = 90^o$




(b) $\Psi = 120^o$




(c) $\Psi = 150^o$

**Figure 4.12:** Instantaneous vorticity contours near the ground plane

was about 75% that of the primary vortex directly above it. This is a weaker secondary vortex than what was observed in simulations involving the impinging vortex ring, described in the previous section. One reason for this difference could be the orientation of the vorticity vector of the primary vortex. In the case of a toroidal vortex ring, the vorticity vector is parallel to the ground plane and the velocity it induces in the ground boundary layer can be significantly higher than that produced by a helical vortex with a skewed vorticity vector as observed in rotorcraft simulations.

Once the instantaneous vorticity contours are computed, the locus of points that have the largest magnitude of vorticity is a reasonable approximation for the vortex trajectory. Figure 4.13 shows the tip vortex trajectory predicted by the hybrid methodology.

The prediction of the free vortex component is shown in green while the prediction of the GPU-RANS solver is shown in red. These predictions are observed to be in reasonable agreement with experimental data.

### 4.2.6 Velocity Magnitudes near the ground plane

The time-averaged and a representative instantaneous velocity fields for the h/R = 1.0 rotor height case are shown in Figs. 4.14(a) and 4.14(b) respectively. The velocities are non-dimensionalized by the ideal inflow in hover ($v_h$) for the corresponding thrust value. Out of ground effect, momentum theory predicts that the maximum velocity in the flowfield should be $2v_h$. In this IGE case the largest

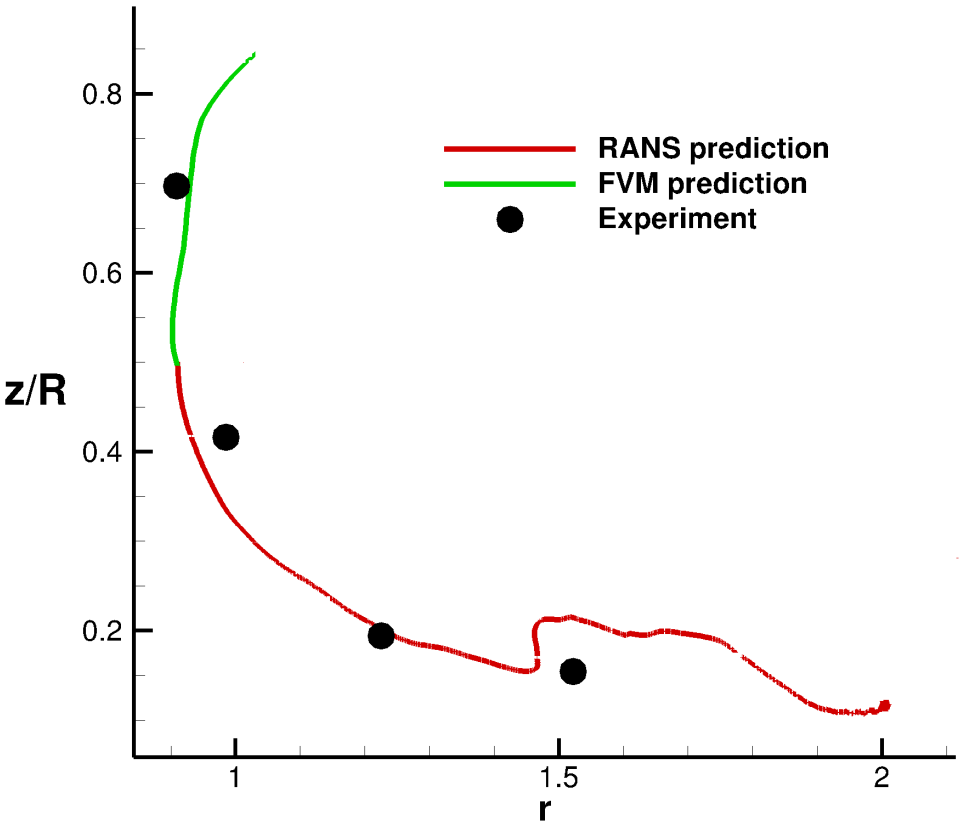
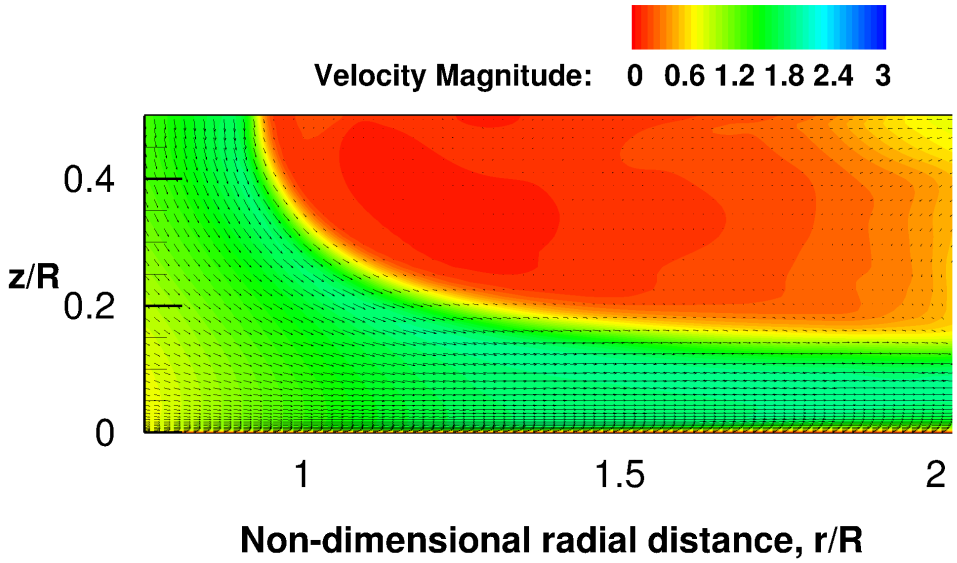
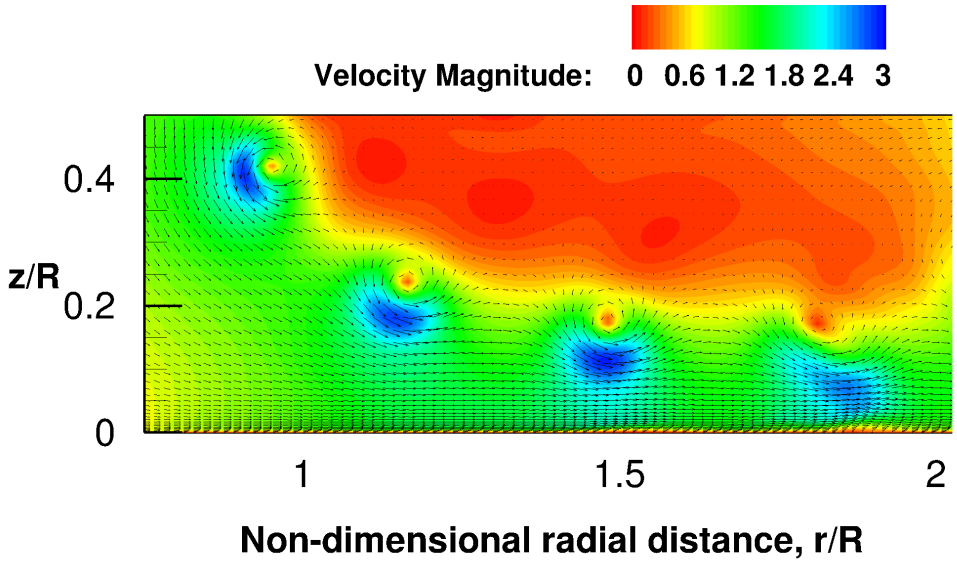



**Figure 4.13:** Hybrid methodology prediction of tip-vortex trajectory compared with experiment




(a) Time-average




(b) $\psi = 0^o$

**Figure 4.14:** CFD predicted time-averaged and instantaneous velocity fields near the ground plane

time-averaged velocities are found to also be around $2v_h$. These maxima in the time-averaged velocity field are seen to occur near the wall for radial stations between r/R = 1.2 to 2.0. Beyond this point, the surface velocities dropped off quickly as the wall jet expanded. Quantitatively, based on continuity considerations, this decrease is expected to be in inverse proportion to the distance from the center of the rotor. This is indeed seen to be the case. While the time-averaged field reveals the structure of the expanding wall-jet, to understand the effect of the passing vortices, one needs to study the instantaneous flowfields. Figure 4.14(b) shows an instantaneous snapshot of the flowfield at a phase angle of $0^o$. Compared to the time-averaged field, the presence of the vortical structures results in regions of the flow seeing even higher flow-velocities than the maximum wall-jet velocity. These velocity perturbations, particularly when the vortex passes close to the ground boundary layer are believed to play a primary role in affecting particle entrainment.

### 4.2.7 Time-averaged velocity profiles

A quantitative validation of the simulated flowfield near the ground plane can be obtained by comparing the time-averaged velocity profiles near the ground with experimental data. Figures 4.15 and 4.16 show the comparison between the experimental and the computed time-averaged radial velocity profiles at four different radial locations (1R, 1.25R, 1.75R and 2R) for the rotor height of 1R above ground.

The velocities are non-dimensionalized by the ideal hover induced velocity

for the corresponding thrust value. From the plot, the computations can clearly be observed to correctly predict the overall physics of the radially expanding wall jet. The height of the wall jet is seen to decrease, both in experiment and simulations, as one moves radially outward. As a consequence, there is corresponding increase in the peak radial velocities with larger radial distances. Comparing the predicted value of peak jet velocity with the experiments, the computed value can be observed to be an over-prediction at outer radial locations. This is expected from the fact that computations predicted much stronger vortices at later wake-ages, inducing stronger velocities at the ground, thereby increasing the time-averaged values.

### 4.2.8 Effect of Additional Trailers

As described in previous sections, the Lagrangian component of the hybrid methodology is formulated on the assumption that the entire vorticity field is confined inside the tip vortex structure. In reality, secondary vortical structures, such as the root vortex or the inboard sheet, are also seen in the flowfield. In steady, hover conditions, these structures can provide a secondary effect on the flow conditions near the ground plane. To understand and quantify this effect, two additional simulations were performed with the following modeling features in the Lagrangian domain:

1. Two trailers - one placed at the tip, one at mid-span. The strength of the second trailer was set to be equal in magnitude and opposite in strength to the tip vortex.

2. Three trailers - one placed at the tip, one placed at one-third span and one

at two-thirds span. The strength of the second trailer was set to be equal in magnitude to the bound circulation at midspan and the strength of the third was set to be equal to the difference between the strengths of the tip vortex and the second trailer.

Figure 4.17 shows the comparison between the experimental and the computed time-averaged radial velocity profiles at two different radial locations (1R, 1.25R) for the rotor height of 1R above ground using one and two trailers.

It is clear that the effect of adding additional trailers serves to increase the wall jet velocities near the ground. This is because the presence of the root vortices causes the tip vortices to descend slightly faster and consequently spend more time closer to the ground inside the RANS mesh. The effect is most pronounced at inboard sections where the presence of additional trailers leads to better correlation with experiment for both wall jet height and maximum velocity magnitudes. The disadvantage to using multiple trailers is the additional computational expense incurred in the free-wake solver due to the doubling of the number of Lagrangian markers.

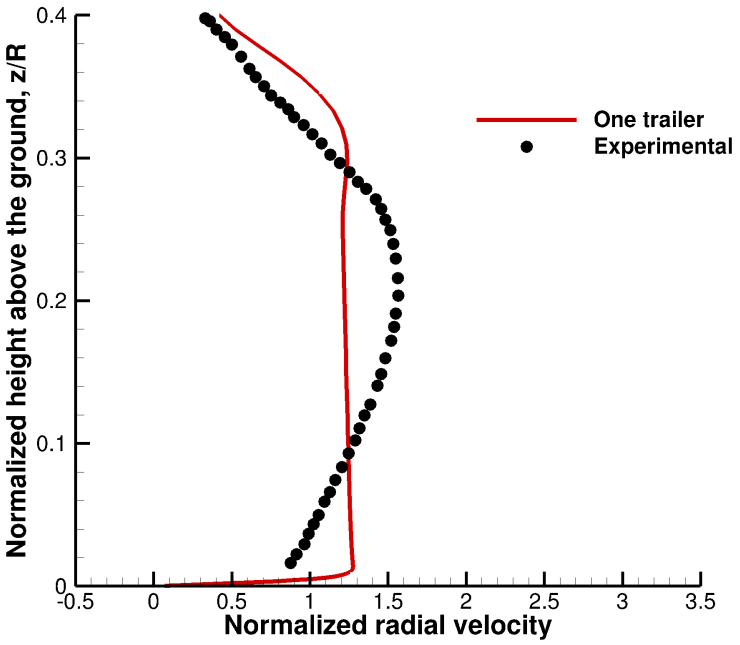
### 4.2.9 Comparison with Full-RANS predictions

In addition to comparing with experimental data, it is also important to compare predictions of the hybrid method with full-RANS simulations. Figures 4.18 and 4.19 compare time-averaged radial velocity profiles predicted by the hybrid methodology and the full-RANS simulation.

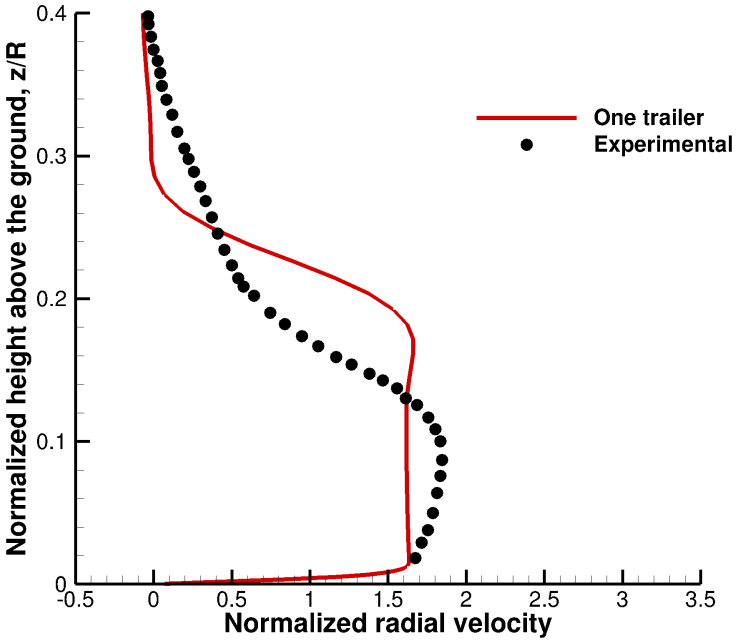
At inboard locations, the full-RANS computations show better agreement with experiment compared to the hybrid method. One possible reason for this

could be that the hybrid method omits the modeling of the inboard sheet while the full-RANS simulation captures the formation and descent of this sheet accurately.

In contrast, at outboard locations, the hybrid method predictions are seen to be in slightly better agreement with experiment compared to full-RANS computations.

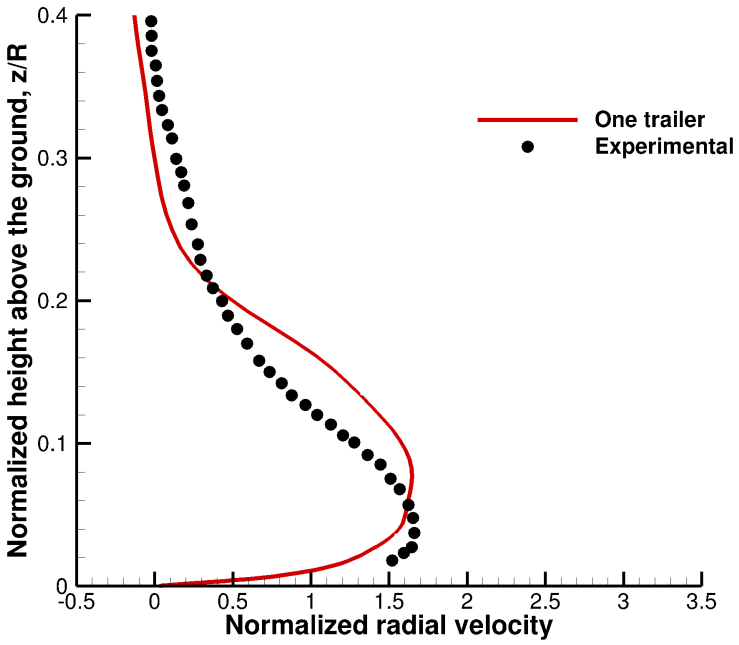



(a) r/R=1.0

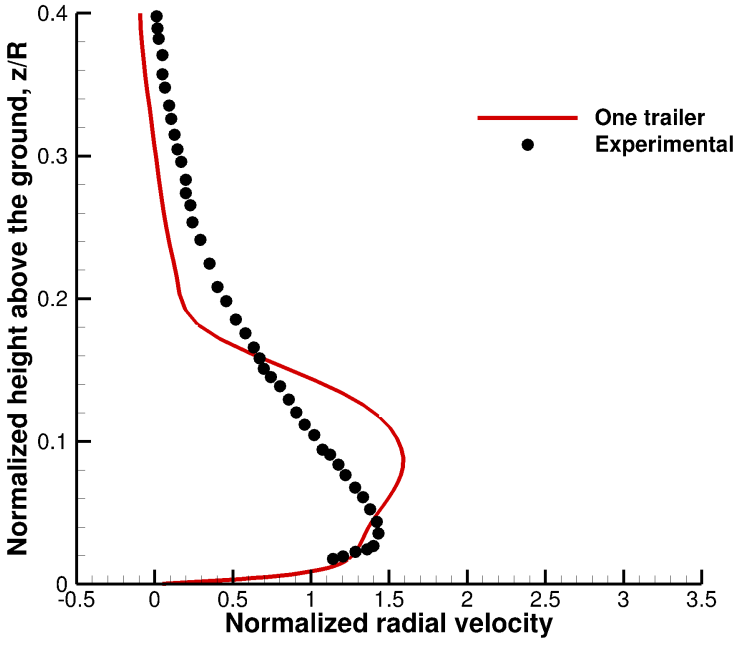



(b) r/R=1.25

**Figure 4.15:** Comparison of computed time-averaged radial velocity profiles with experiment

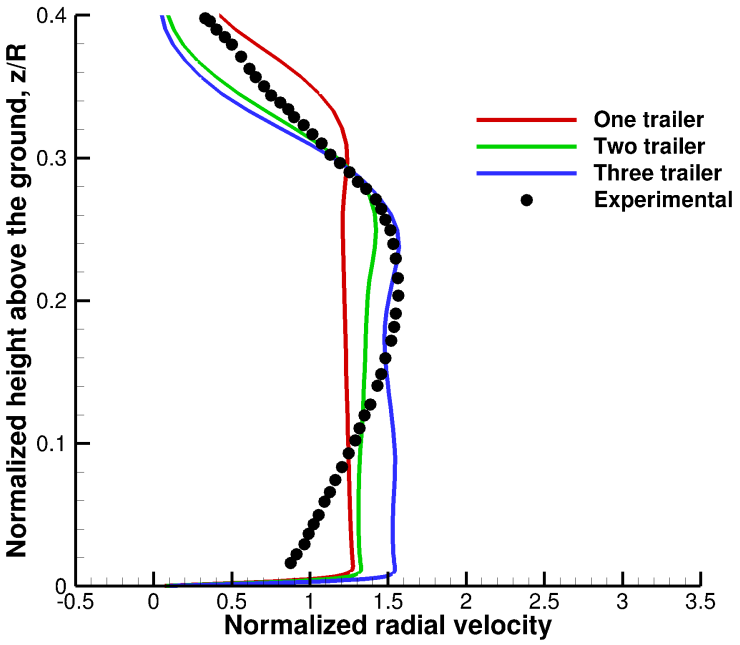



(a) r/R=1.75

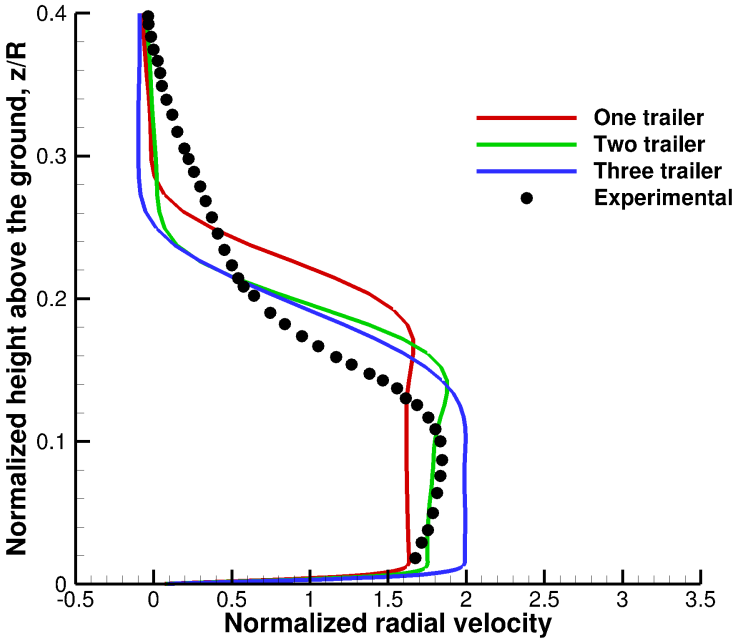



(b) r/R=2.0

**Figure 4.16:** Comparison of computed time-averaged radial velocity profiles with experiment

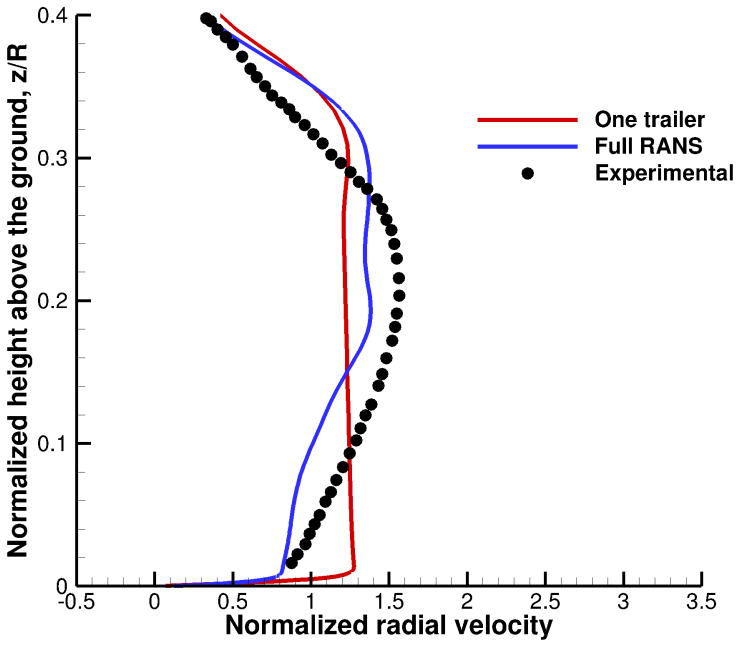



(a) r/R=1.0

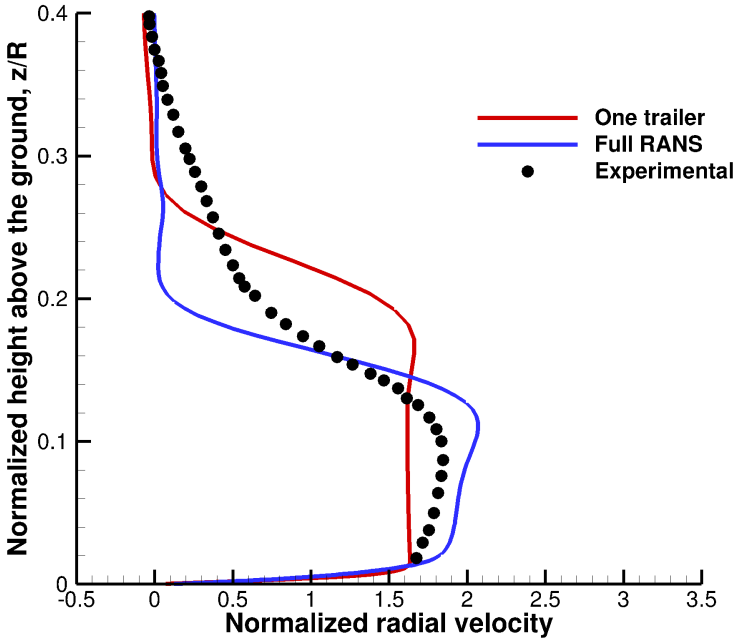



(b) r/R=1.25

**Figure 4.17:** The influence of additional trailers on CFD predicted time-averaged velocity profiles near the ground plane

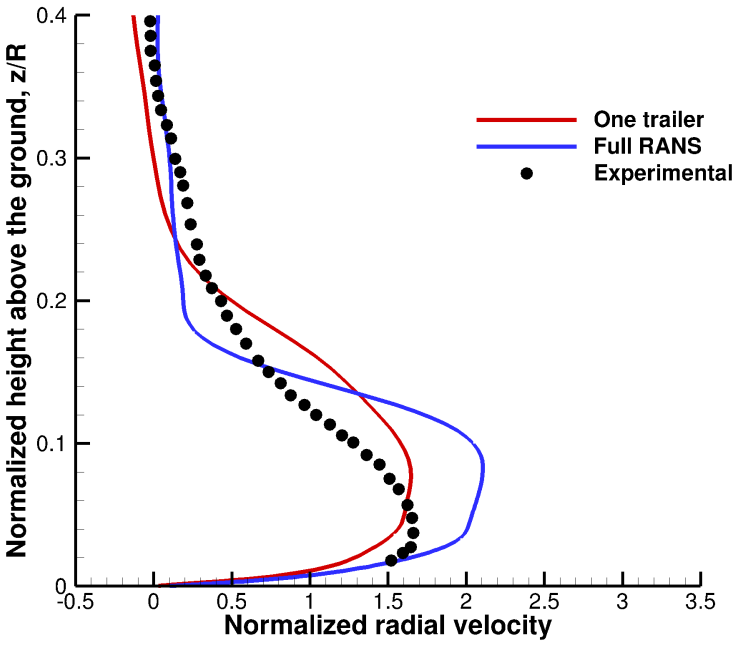



(a) r/R=1.0

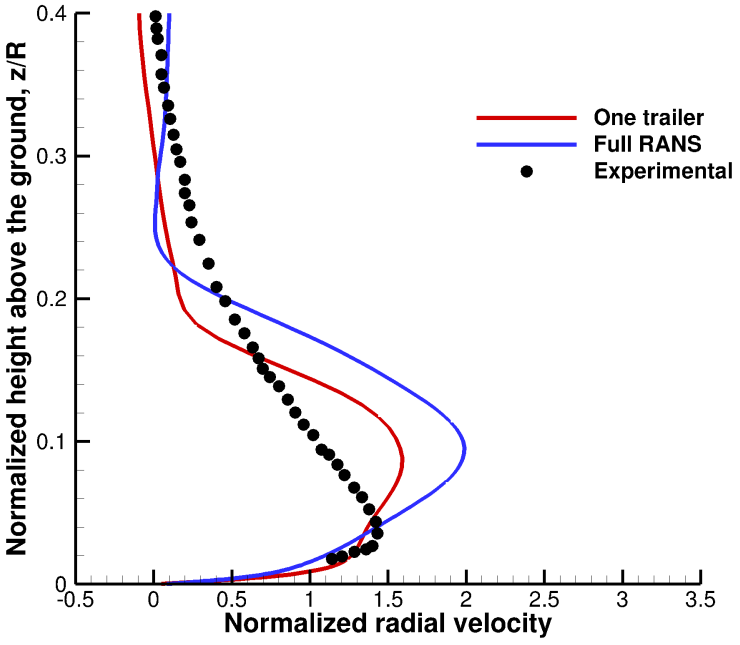



(b) r/R=1.25

**Figure 4.18:** Comparison of predictions by the hybrid method and full-RANS computations of time-averaged velocity profiles at the ground plane




(a) r/R=1.75

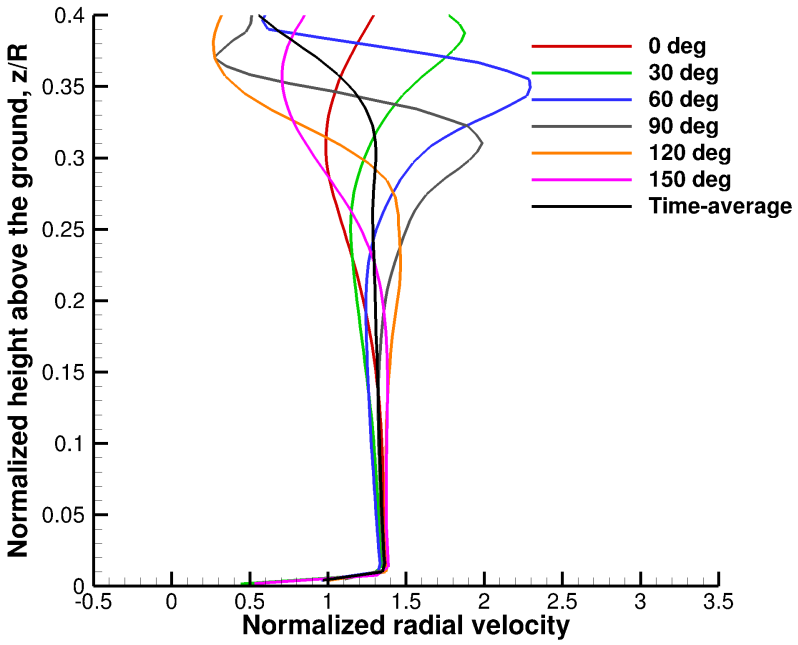



(b) r/R=2.0

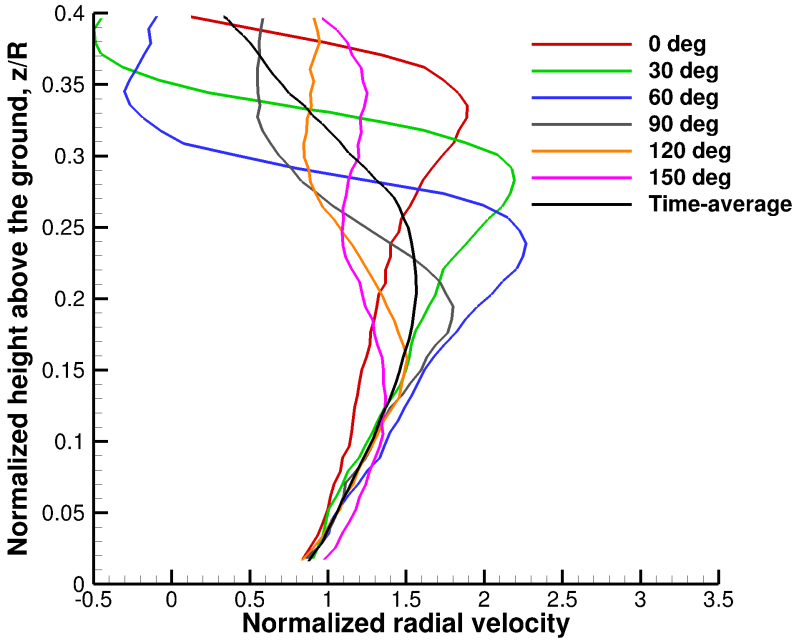
**Figure 4.19:** Comparison of predictions by the hybrid method and full-RANS computations of time-averaged velocity profiles at the ground plane

### 4.2.10 Phase-averaged velocity profiles

A comparison of velocity profiles at different wake-ages (phase-averaged) is plotted for various radial distances (r/R = 1.0, r/R = 1.25, r/R = 1.75, r/R = 2.0) in Figs. 4.20, 4.21, 4.22 and 4.23 between the CFD simulations and the experimental measurements. Plotted along with this are the time-averaged radial velocity profiles, which were discussed earlier. The radial velocity profiles are non-dimensionalized by the ideal hover induced velocity for the corresponding thrust value. The fluctuations seen in the phase-averaged profiles at each radial location represent the velocity signature of a passing vortex. These fluctuations are indicative of the unsteadiness in the flowfield as a function of wake age. At all the radial locations, the time-averaged velocity is equal to the average of all the phase-averaged velocities. The fluctuations do not show up in the time-averaged values due to the averaging out of the transients. Again, due to the stronger predicted vortex strength, the fluctuations in the computed phase-averaged velocities are larger, especially at outboard radial locations as compared to those measured experimentally. In addition, there seems to be a slight vertical offset in the position of the passing vortices at these radial locations.

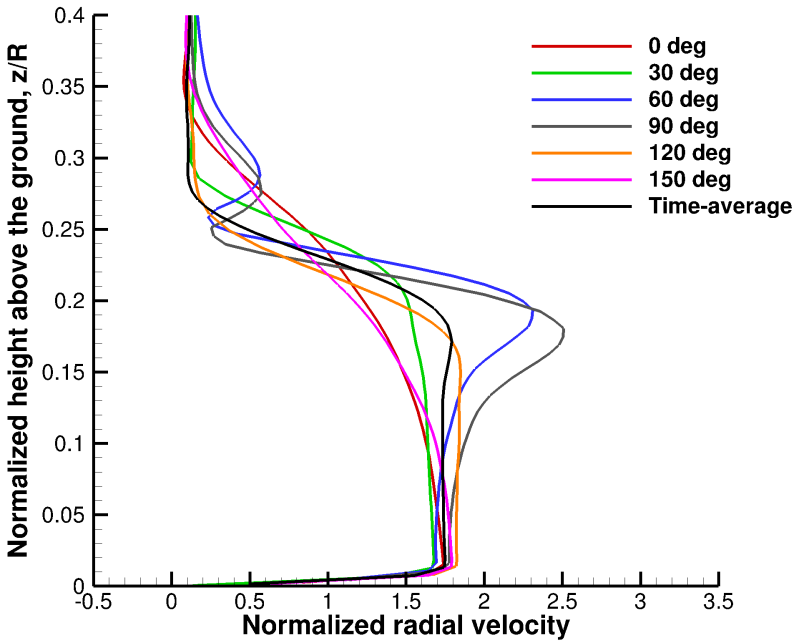



(a) r/R=1.0 (CFD)

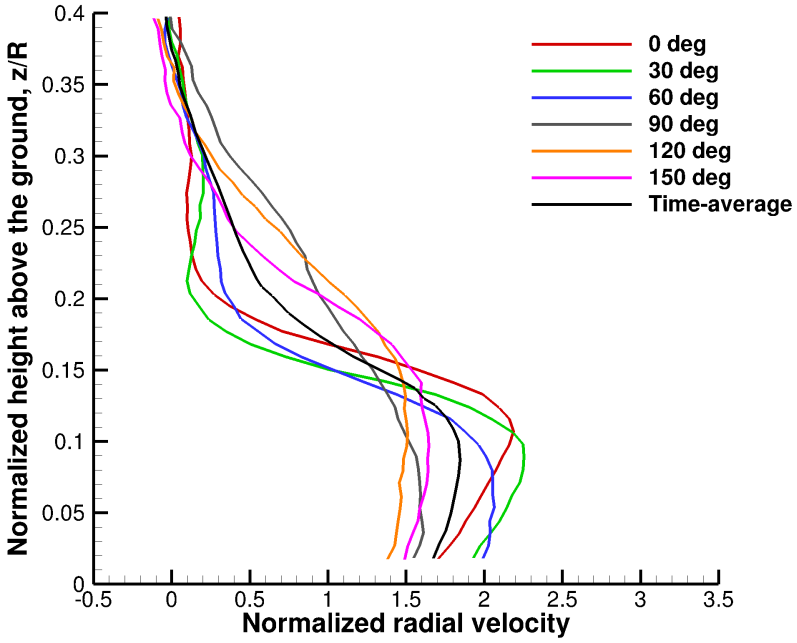



(b) r/R=1.0 (Experiment)

**Figure 4.20:** Comparison of computed phase-averaged radial velocity profiles with experiment

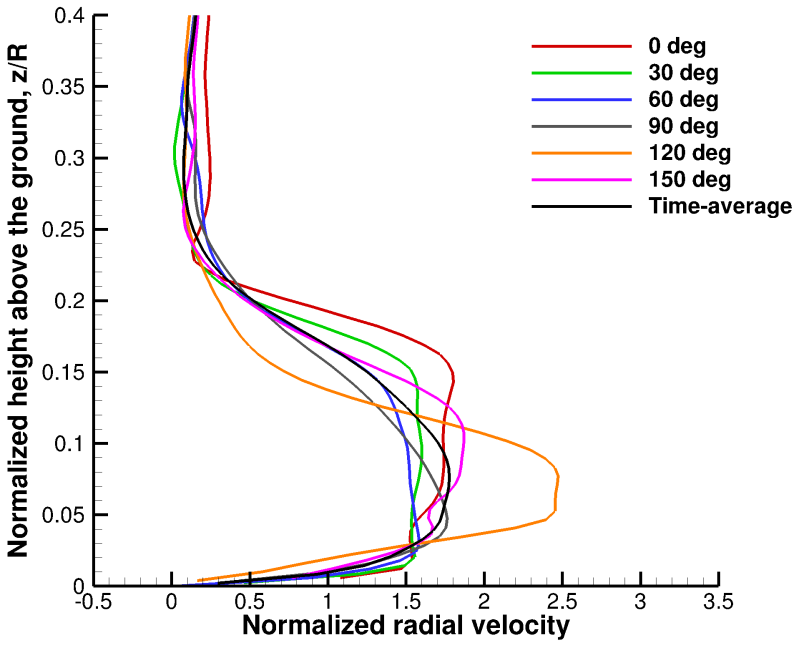



(a) r/R=1.25 (CFD)

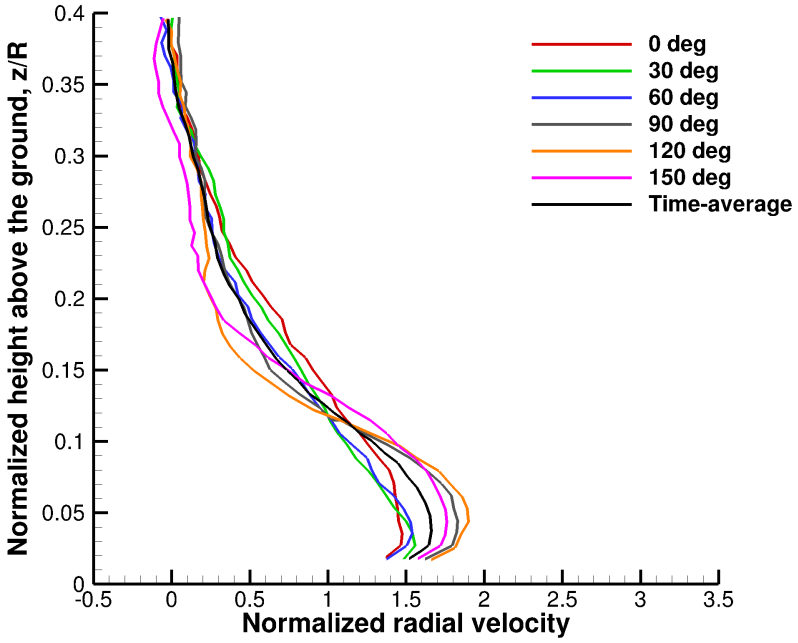



(b) r/R=1.25 (Experiment)

**Figure 4.21:** Comparison of computed phase-averaged radial velocity profiles with experiment

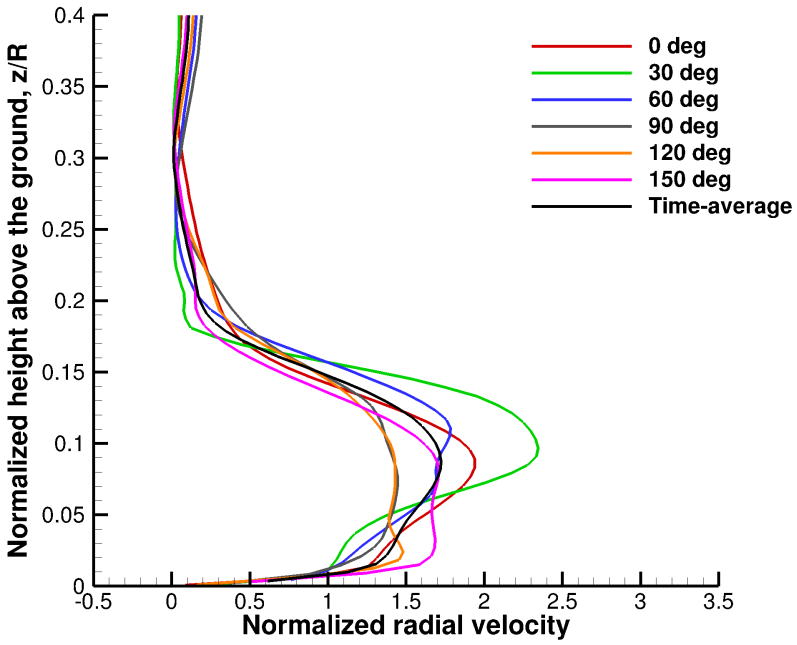



(a) r/R=1.75 (CFD)

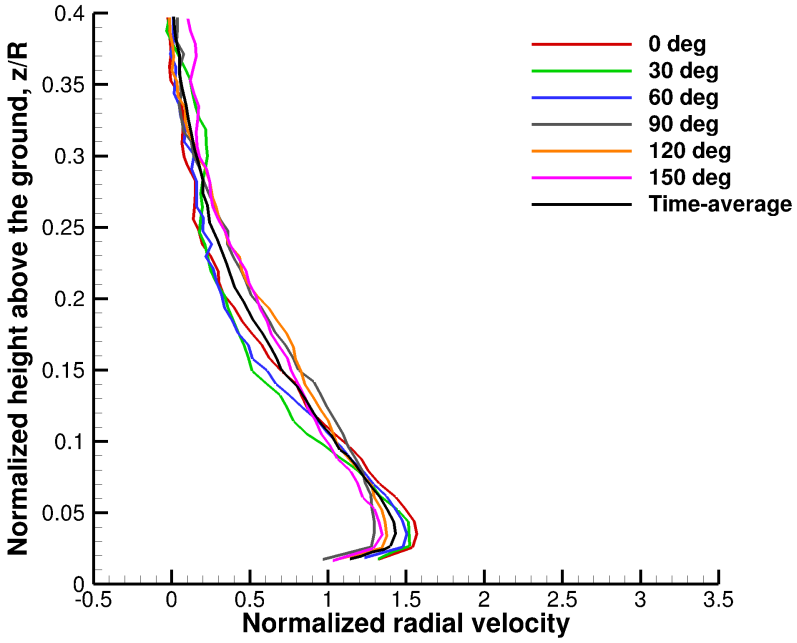



(b) r/R=1.75 (Experiment)

**Figure 4.22:** Comparison of computed phase-averaged radial velocity profiles with experiment




(a) r/R=2.0 (CFD)




(b) r/R=2.0 (Experiment)

**Figure 4.23:** Comparison of computed phase-averaged radial velocity profiles with experiment

### 4.2.11 Eddy viscosity contours at the ground plane

Eddy viscosity contours can be used to better view the turbulence levels at the ground. Figures 4.24 show the eddy viscosity contours (normalized by laminar viscosity) at the ground plane for a rough wall. The smooth wall model sets the turbulent stresses to zero at the ground plane as a boundary condition. In contrast, the rough wall correction [59] allows for non-zero turbulent stresses at the ground plane. For the rough wall result shown, the roughness height was

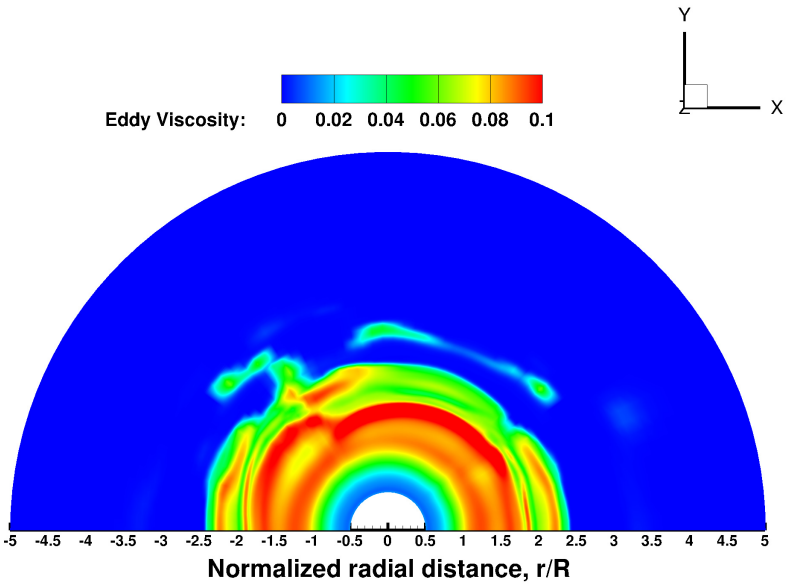



**Figure 4.24:** Predicted eddy viscosity contours at the ground plane using a rough wall simulation

chosen to be three times the diameter of a 100 micron particle based on trends described in [83]. As can be seen, the turbulence levels are higher in the case of the rough wall. These higher values of eddy viscosity translate to larger effective shear stresses at the ground. Figure 4.25 shows the friction velocity distribution across the ground plane as predicted by a rough wall simulation. Compared to

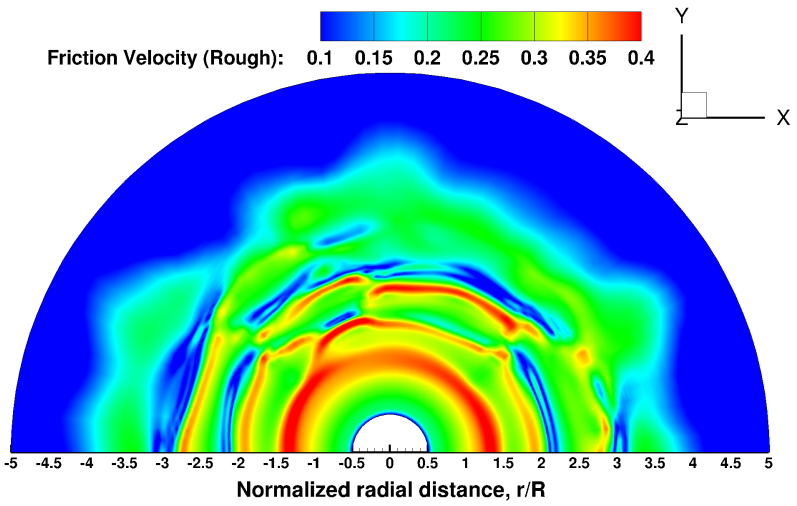



**Figure 4.25:** Computed friction velocity contours across the ground plane using a rough wall simulation

the smooth wall simulation, the friction velocity is found to be as high as 15% more than the corresponding friction velocity in a smooth wall simulation. This additional shear at the ground plane could lead to particles of larger diameters being mobilized or entrained.

### 4.2.12 Brownout Cloud Simulation

Having validated the accuracy of the CFD predicted aerodynamic flowfield, the brownout cloud is simulated by coupling the hybrid model with a Lagrangian particle solver. To simulate the cloud, a circular sediment bed of radius equal to 4R was placed directly below the micro-rotor at a rotor-height of $1R$. The number of layers in the sediment bed was set to 20.

Three different particle sets were used in the simulations to mirror the exper-

imental work conducted by Sydney et al. [16]. In Ref. 16, three different sets of spherical glass beads having density of $1300 kg/m^3$, but varying in diameter were used to study the effect of particle size on the phenomenological attributes of the brownout cloud. The particle sizes from the experiment and the simulation are shown in Table 4.1.

| Particle Set | Diameter (microns) | Density ($kg/m^3$) |
|---|---|---|
| A (heavy) | 100 | 1300 |
| B (intermediate) | 50 | 1300 |
| C (light) | 1 | 1300 |

**Table 4.1:** List of particle parameters used in the coupled simulations

Table 4.2 shows the threshold friction velocities for the three different particle sets predicted by the Bagnold entrainment model.

| Particle Set | Diameter (microns) | Threshold friction velocity ($m/s$) |
|---|---|---|
| A | 100 | 0.24 |
| B | 50 | 0.31 |
| C | 1 | 2.07 |

**Table 4.2:** Threshold friction velocities for the three different particle sizes

### Particle Set A

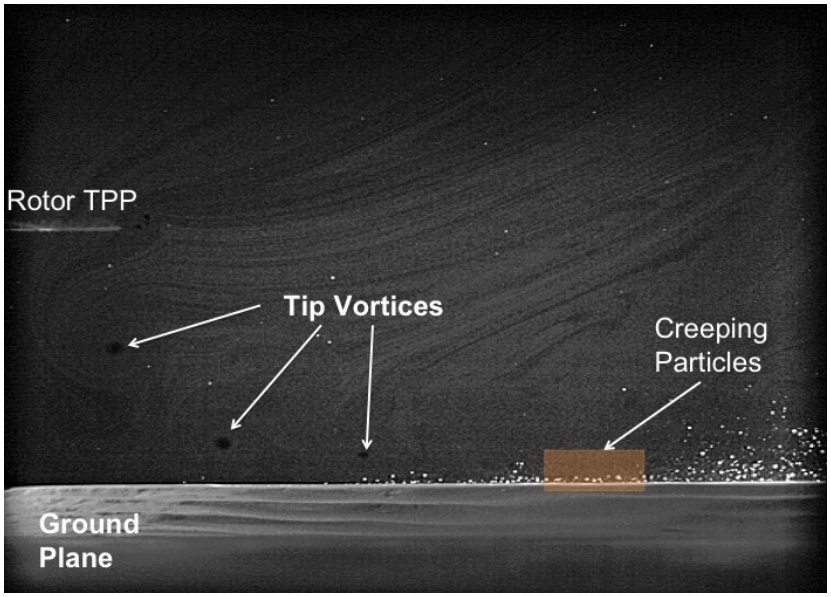
In simulations involving particle set A, the particles were observed to simply creep along the ground plane in a radial direction without ever uplifted into suspension. Figure 4.26(b)-(j) are nine consecutive instances (with focus on two specific beads) from a simulation showing the process by which particles in set

A creep along the ground. For the sake of clarity, only a single layer in the sediment bed is modeled. Similar observation were reported by Sydney et al. [16] in experiments with heavy glass spheres (see Fig.4.26(a)). The heaviest particles were found to simply roll or hop along the surface and very few were uplifted into suspension. One explanation for this observation is that the larger inertia associated with this particle set effectively makes each particle a low-pass filter that responds primarily to the radially-outward, time-averaged wall jet and not the high-frequency perturbations associated with vortical flow.

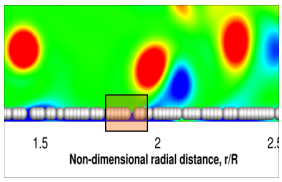
**Particle Set B**

An important mechanism observed in the numerical experiments with particle set B is vortex trapping. Particles are uplifted by vortices that come into close proximity to the ground plane. These particles are then entrained into subsequent vortices. These particles are then said to be 'trapped' inside these vortices. Figures 4.27(b)-(j) are nine consecutive instances from a simulation showing the process by which a set of particles get trapped by a passing vortex. The mechanism of vortex trapping was reported by Sydney et al. [16] to be one of the key mechanisms by which particles are lifted into suspension. Figure 4.27(a) shows flow visualization for one instance of the vortex trapping phenomenon as seen in experiment. For the sake of clarity, only a single layer in the sediment bed is modeled. The results from the numerical simulations are in qualitative agreement with the experimental data.

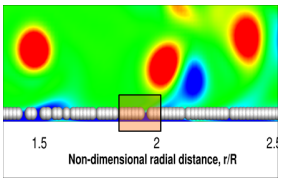
Figure 4.28(a) compares a snapshot of the dual-phase simulation with an image visualizing the particle distribution seen in the laboratory experiments (Fig. 4.28(b)) conducted by Sydney et al. [16]. Sediment waves, outboard of



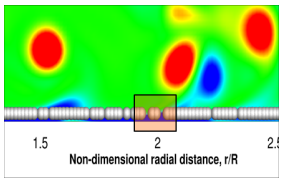

(a) The mechanism of creep observed in experiments



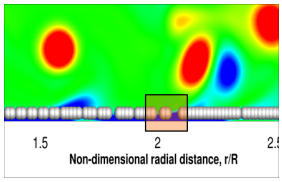

(b) Instance 1



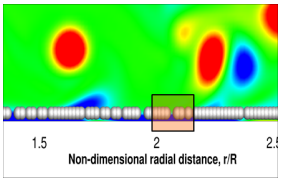

(c) Instance 2



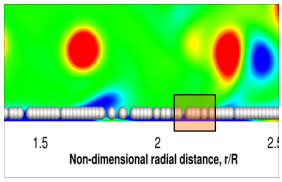

(d) Instance 3



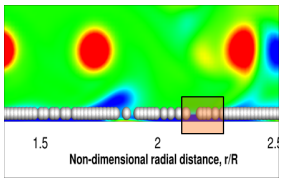

(e) Instance 4



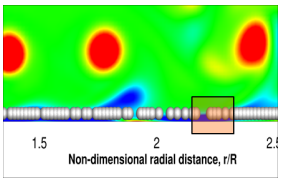

(f) Instance 5



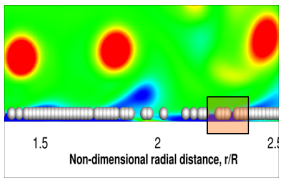

(g) Instance 6




(h) Instance 7

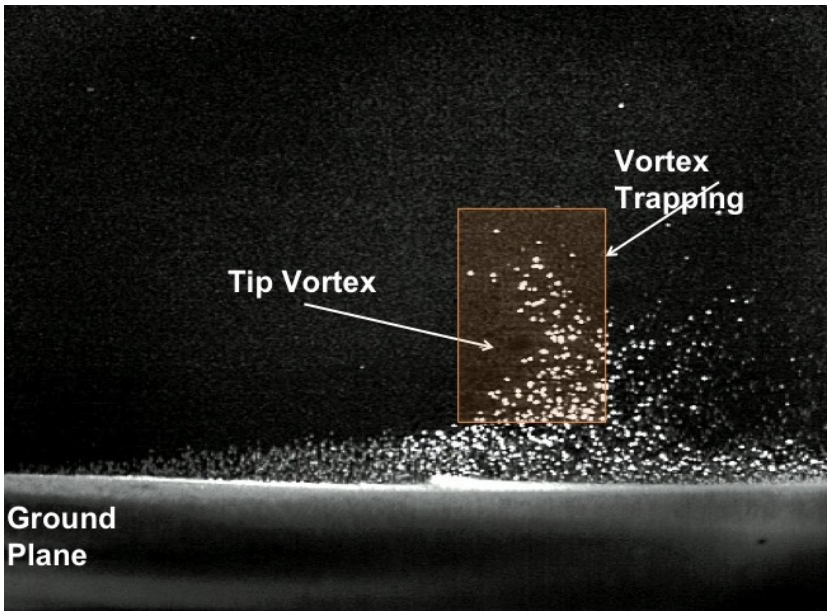



(i) Instance 8

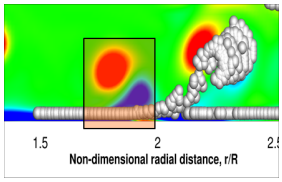



(j) Instance 9

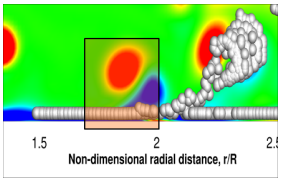
**Figure 4.26:** The mechanism of creep seen in experiment and simulations with a single layer of particle set A. Note: The particles are not drawn to scale.



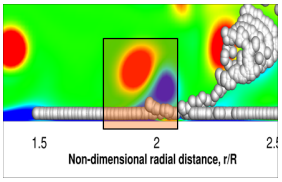

(a) The mechanism of vortex trapping observed in experiments



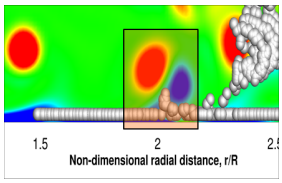

(b) Instance 1



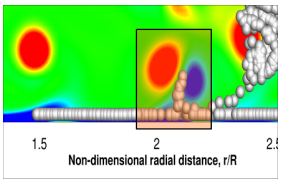

(c) Instance 2



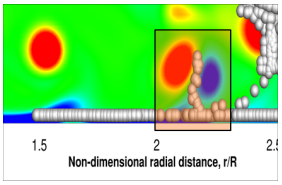

(d) Instance 3



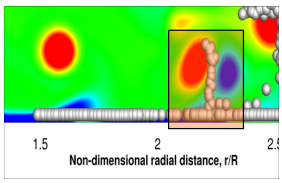

(e) Instance 4



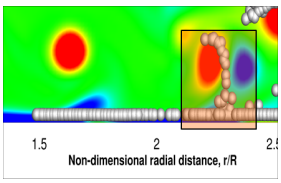

(f) Instance 5



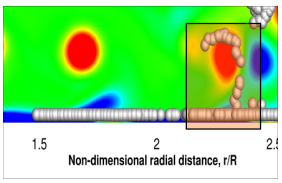

(g) Instance 6




(h) Instance 7



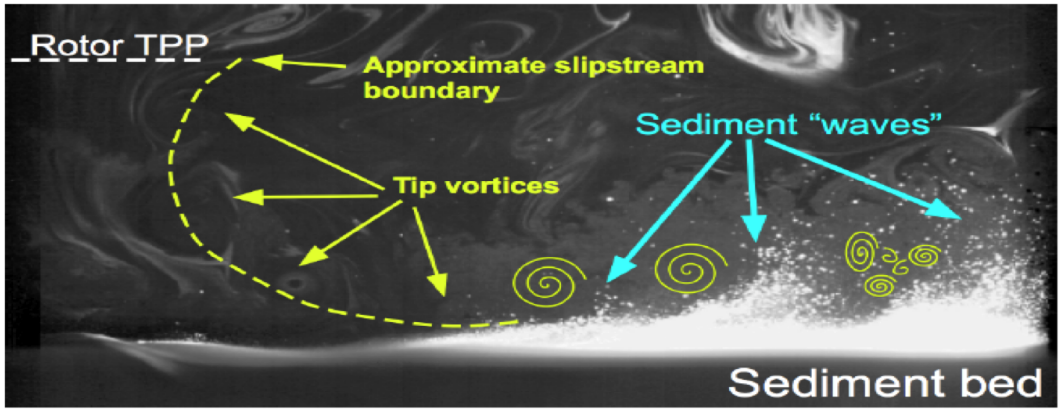

(i) Instance 8



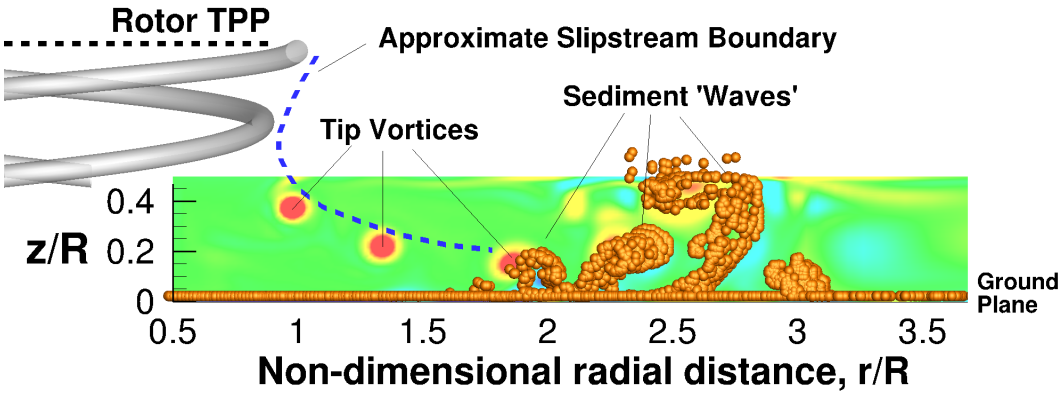

(j) Instance 9

**Figure 4.27:** The mechanism of vortex trapping seen in experiment and simulations with a single layer of particle set B. Note: The particles are not drawn to scale.

the rotor, similar to those seen in experiments, are captured in simulations if we allow for multiple layers of particles in the sediment bed.



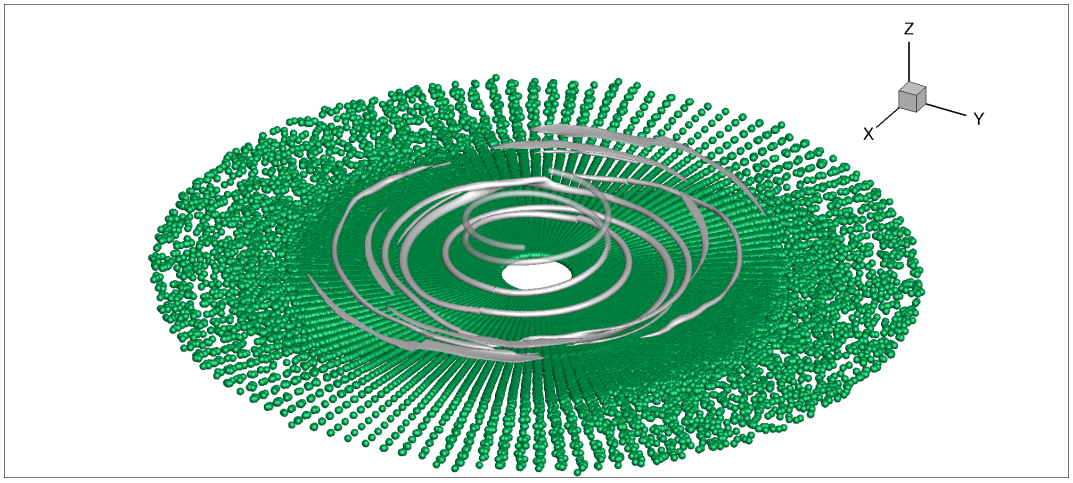

(a) Formation of sediment waves seen in experiment



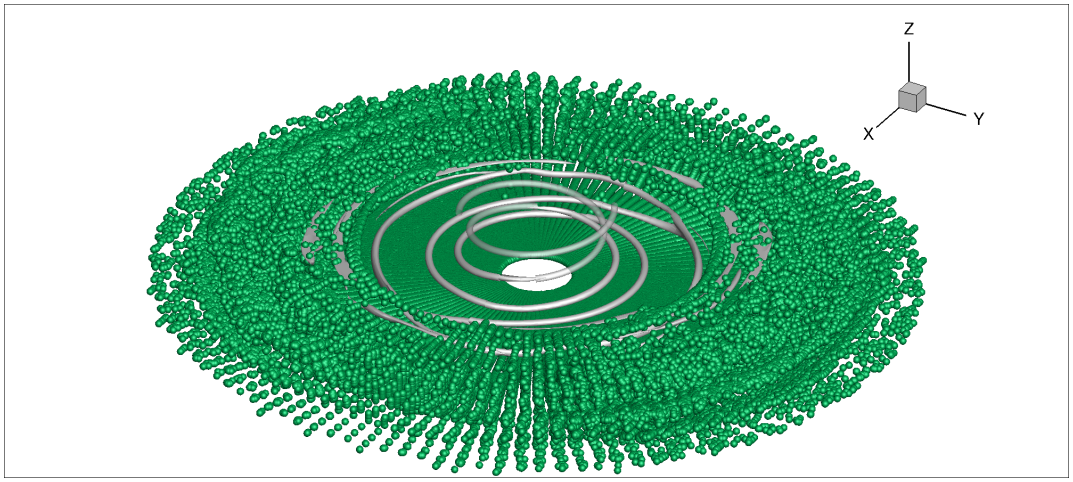

(b) Formation of sediment waves seen in simulations

**Figure 4.28:** The formation of sediment waves outboard of the rotor. Note: The particles are not drawn to scale.

Figure 4.29 is an isometric snapshot of a dual-phase simulation showing the 3D structure of a typical brownout cloud. Clearly defined and sustained three-dimensional sediment waves are seen to form in simulations involving particles from set B.




(a) Isometric view of the dual-phase flowfield near the start of the computations. Both the free-vortex wake away from the ground and iso-surfaces of vorticity inside the RANS mesh are plotted to show the tip vortex trajectory. Most particles are lying immobile on the ground plane.




(b) Isometric view of the dual-phase flowfield after 16.25 revolutions. The dispersed phase is highly active with particles being entrained around close-proximity vortices to form well defined three-dimensional waves.

**Figure 4.29:** Predicted 3D structure of the dual-phase flowfield with 50 micron particles beneath a hovering MAV-rotor. Note: The particles are not drawn to scale.

### Particle Set C

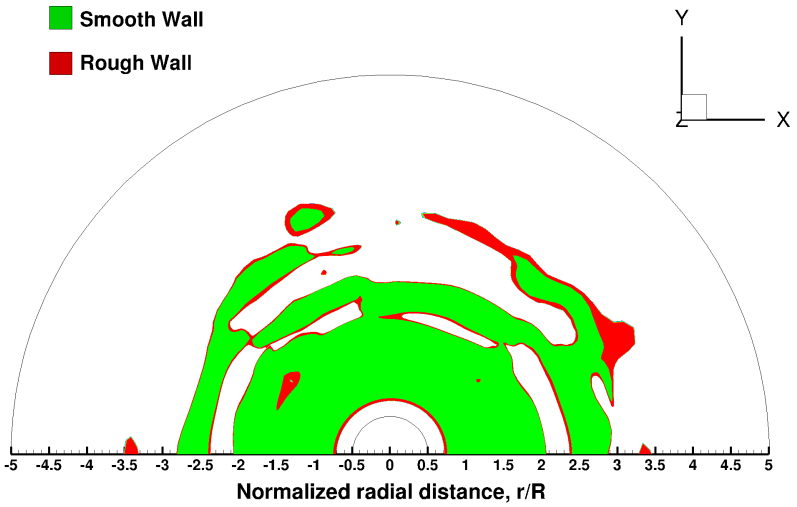
The simulations involving particle set C predicted an immobile sediment bed with no particles being uplifted into suspension. This is because the entrainment

model used in the sediment tracking algorithm predicts a very high threshold friction velocity needed for the particles in set C to become mobile owing to the increased inter-particle cohesive forces at this scale. This threshold exceeds the maximum friction velocity observed at the sediment ground plane and hence no particle mobilization is seen in these simulations. These results are similar to the observations made in the experiments by Sydney et al. [16] where the lightest particles were the least mobile due to strong cohesive forces between particles of small sizes.

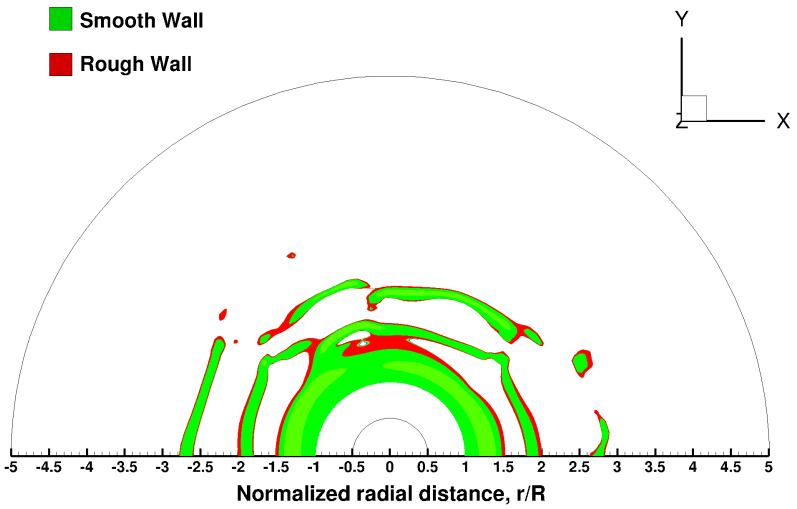
**Effect of Surface Roughness on Particle Mobilization**

The addition of the roughness correction to the SA model increases the turbulent stresses at the ground plane and consequently the average friction velocity. Fig 4.30(a) shows the regions on the ground plane predicted to have mobile 100 micron particles. The region in green depicts predictions from the smooth wall simulation. The region in red shows predictions from the rough wall simulation. Clearly, the rough wall simulation predicts a slightly more mobile sediment bed.

Fig 4.30(b) shows the regions on the ground plane predicted to have mobile 50 micron particles. The region in green depicts predictions from the smooth wall simulation. The region in red shows predictions from the rough wall simulation. Again, as in the case of the 100 micron particles, the rough wall simulation predicts a slightly more mobile sediment bed.




(a) Regions of mobilization for the 100 micron particles




(b) Regions of mobilization for the 50 micron particles.

**Figure 4.30:** Regions of the ground plane predicted to have sufficient friction velocity to mobilize particles

### 4.2.13 Timing and Profile Data

Figure 4.31 illustrates the computational expense of the various algorithmic components of the hybrid solver when it is run on the GTX Titan. One key difference

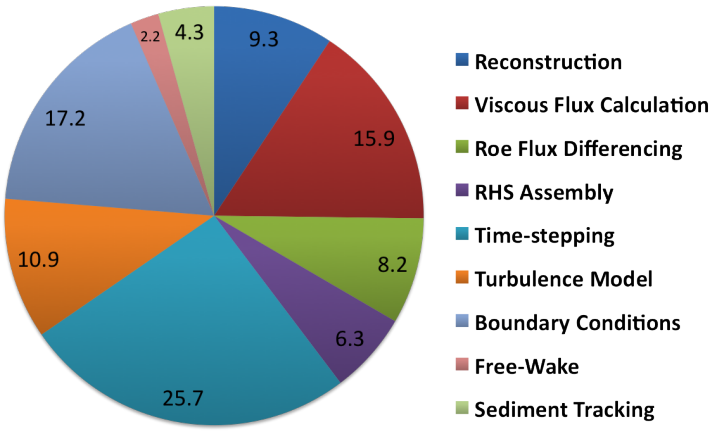



**Figure 4.31:** Profile data on the GTX Titan for the hovering micro-rotor simulation

between this test-case and previous benchmarking results presented in Chapter 3 is the significant proportion of GPU resources allocated to the computation of boundary conditions. This is because the boundary conditions now involve the calculation of velocities induced by the free-wake filaments on the RANS boundary nodes. This computation scales with the product of the number of filaments and the number of boundary elements, i.e. the complexity of this calculation is $O(nm)$ where n is the number of vortex filaments and m is the number of field points. For such calculations, the parallelism of GPU architectures coupled with their higher compute power makes GPU platforms significantly superior to their CPU counterparts. This superiority is reflected in the following timing comparison: On the GTX Titan, the hybrid solver took about 1.25 hours to complete

one rotor revolution. In contrast, a hybrid methodology implementation using the domain decomposition paradigm [84] took 6 hours to complete one revolution when run on 32 Intel Xeon cores.

## 4.3 Summary

In this chapter, time-accurate computations of single and dual-phase brownout environments were simulated using a hybrid methodology and validated with available experimental data. The computed flowfields were examined to extract relevant flow features and to better understand the underlying flow physics. Two experimental studies were used as validation databases:

1. Single-phase impinging vortex ring [17]
2. Single and dual-phase studies of a two-bladed, hovering micro-rotor [16]

Listed below are the specific observations/conclusions from the simulations of these two experimental setups:

1. The 3D GPU-RANS solver was coupled to GPU-based free-wake solver to simulate the single-phase flow environment around an impinging vortex ring. Predictions using this hybrid methodology were shown to be in good agreement with experimental data.
2. The comparison of predicted time-averaged radial velocity profiles with experimental data showed good agreement at all radial locations where the data was available. In particular, the peak wall-jet velocities at these radial locations were found to be well predicted.

3. Isocontours of vorticity revealed the formation of secondary vortical structures when the vortex ring came into close proximity to the ground was observed in simulations. The size, strength and position of these secondary structures was estimated and found to be in qualitative agreement with observations made in experiments by [81].

4. The 3D GPU-RANS solver was coupled to GPU-based free-wake solver and a sediment tracking algorithm to simulate the single and dual-phase flow environment beneath a hovering MAV-scale rotor. Predictions using this hybrid methodology were shown to be in good agreement with experimental data.

5. The phase-averaged azimuthal vorticity contours was found to be in reasonable agreement with experimental results in terms of wake trajectory and vortex strength.

6. The comparison of predicted time-averaged radial velocity profiles with experimental data showed good agreement at all radial locations where the data was available.

7. Friction velocity patterns at the ground plane show that the maximum magnitudes of shear are seen at radial locations between 1.5R and 2.5R. The local induced velocity of the tip vortex closest to the ground serves to energize the boundary layer leading to an increase in shear stress at the ground plane. This suggests that in brownout conditions, maximum sediment bed mobility can be expected in this region.

8. The brownout simulation showed that the heaviest particles (100 microns)

mostly crept along the ground plane without being uplifted into suspension. This is in agreement with experimental data.

9. The lightest particles (1 microns) experienced no motion whatsoever; a result similar to that reported in Ref. 16 wherein experiments with the lightest particles resulted in the least mobile bed due to cohesive forces between particles that dominate at this end of the size spectrum.

10. The intermediate sized particles (50 micron) were found to be the most mobile under the given conditions, exhibiting most of the phenomena observed in laboratory experiments conducted by Sydney et al. [16]. The mechanisms of creep and vortex trapping were observed to occur during the simulation. Also, similar to observations in experiments, the creation of sustained sediment waves outboard of the rotor was seen to occur.

11. The use of the roughness correction in the turbulence model led to predictions of higher turbulent stresses at the ground plane and consequently a slightly more mobile sediment bed for both particle set A and particle set B.

12. The GPU-based hybrid method has a memory footprint of around 900MB per million grid points and a run-time of approximately 1.25 hours/rev. In comparison, the hybrid method based on the domain decomposition paradigm, described in [84] has a run-time of approximately 6 hours/rev when run on 32 Intel Xeon cores.

# 5

# Conclusions

The phenomenon of brownout is a serious operational concern for rotorcraft operating in ground effect. The developing cloud has the potential to cause visual obscuration, disorienting the pilot and adversely affecting his ability to operate the vehicle safely. It is therefore desired to have a detailed understanding of the flow physics that leads to the creation of brownout conditions. The work reported in this dissertation attempts to develop and validate a GPU-based, numerically efficient, high resolution computational methodology that can be used to study problems of this nature in great detail. This methodology was designed to be hybrid, combining the numerical efficiency of a free-vortex method with the relative high-fidelity of a RANS solver.

This final chapter summarizes the contributions made in this work, and lists the main observations and conclusions drawn from the study. Suggestions for future research are also provided.

## 5.1 Summary

The overall objective of this dissertation was to develop and validate a numerically efficient, high resolution computational methodology to study the single and dual-phase flow physics of brownout environments. This required the design and development of a multi-granular, compressible Reynolds-Averaged Navier Stokes (RANS) solver. The individual algorithms constituting the RANS solver were systematically verified and validated for simpler problems before being integrated into the hybrid methodology. The hybrid methodology was then applied to simulate conditions similar to those encountered in rotorcraft brownout.

The first test case was an impinging vortex ring based on the experiments performed by Geiser [81] and Mulinti [17]. The computations were validated by comparing predicted time-averaged, velocity profiles at the ground plane with experimental measurements. Further, qualitative, validation was achieved by the prediction of the formation of a secondary vortical structure at the ground plane - a phenomena reported in experiments. The approximate position, size and strength of this secondary vortex, relative to the primary vortex were also found to be in good agreement with experimental observations.

Next, the hybrid methodology was applied to the dual-phase simulation of a micro-scale rotor hovering in ground effect, based on the experiment by [16]. The single-phase computations were validated by comparing the predicted time-averaged and phase-averaged velocity profiles at the ground plane with experimental measurements. The predictions were found to be in reasonable agreement

with experimental data. Furthermore, a comparison of the predicted tip-vortex trajectory with experiment showed good qualitative agreement. In addition to validating the hybrid methodology, a detailed study of the flow physics was performed. Visualization of the computational results showed the intermittent formation of secondary vortical structures near the ground plane, similar to, but relatively weaker and smaller than those observed in the impinging vortex ring case.

After gaining sufficient confidence in the single-phase simulation of the hovering micro-rotor, a dual-phase simulation of the same experimental setup was performed. A sediment bed was placed inside the computational model, at the ground plane and individual particles were coupled, unidirectionally, to the single-phase flow. Three separate particle sizes were studied, corresponding to the experiments by Sydney et al. Each particle set revealed a unique response to the forcing aerodynamics. The mechanisms of creep and vortex trapping were observed in simulations. The variation of the dominant transport mechanisms as a function of particle size was also found to be in accordance with experimental observation. In addition, predicted friction velocity patterns at the ground plane showed that the maximum magnitudes of shear are seen at radial locations that correlated well with increased sediment bed mobility, both in simulations as well as experiment, providing further validation to the entrainment model used in the sediment tracking algorithm [1] employed in this work.

## 5.2 Observations

Specific observations and conclusions drawn from the development and use of the GPU-RANS solver and hybrid methodology are detailed below.

### 5.2.1 GPU-RANS Solver

1. The GPU-RANS solver was found to be very efficient when run in fine-grain mode - for example, when each thread is mapped to a single grid-cell. Any deviation from the fine-grain mode of parallelism was observed to be accompanied with a performance penalty. For instance, the inversion of an implicit system using a non-iterative algorithm requires that a single thread operate on multiple grid cells. If the number of grid cells operated upon by a single thread is a small fraction of the total number of grid cells, then this mode of parallelism can be considered an intermediate coarse grain parallelism. The granularity of parallelism increases with the number of grid cells operated upon by a single thread - the coarsest granularity of parallelism corresponds to a single thread being active during an operation.

2. Explicit methods in RANS solvers are most amenable to a fine-grain mode of parallelism. It was shown using the canonical test case of a shock-vortex interaction that if only explicit algorithms are employed in a RANS simulation, GPU platforms can provide large speedups ( 50X) over equivalent serial computations. One disadvantage with explicit methods is that they typically impose strict restrictions on the level of discretization that the solver uses. For instance, an explicit time-stepping scheme is constrained

by the CFL condition. This restriction becomes even more stringent when an attempt is made to resolve lengths of vastly different scales, as is done in viscous, turbulent simulations at large Reynolds numbers. For explicit methods to be employed in such simulations would require the use of very small time-steps, making the overall runtime for a simulation computationally prohibitive.

3. Unlike explicit methods, implicit methods are not accompanied by stability-related, discretization restrictions. However, the disadvantage with implicit methods encountered in RANS solvers is that the solution procedure typically involves the inversion of a linear system which couples together several grid-cells precluding the use of a fine-grain approach in this step. This deviation from fine-grain parallelism can cause significant performance penalties, particularly if a large number of grid-cells (and the variables contained within them) are coupled together. In the extreme case, a direct inversion of a system, equal in size to the grid, would necessarily be a serial process and consequently, a highly inefficient bottleneck.

4. Fortunately, there exist a variety of options between explicit methods and direct inversion of large linear systems. One such family of methods are referred to as line-implicit methods where the large linear system comprising the entire grid is factored into a set of smaller implicit systems, with each system coupling grid cells that lie along a single coordinate line. Such schemes are quite popular in CFD literature and are often used for operations ranging from time-stepping (DADI) and spatial reconstruction (CR-WENO) to turbulence and transition modeling (DDADI). The line-parallel

nature of these approaches allows for an intermediate grain of parallelism where each thread is mapped to a single coordinate line. Therefore, if the number of such parallel and independent systems is sufficiently large (as is typically the case in 3D), GPU resources can be fully utilized and the performance penalty associated with implicit methods can be reduced significantly on such platforms. At the same time, the advantageous characteristics of such methods such as improvements in stability bounds, higher orders of accuracy, etc., can be exploited.

5. In Chapter 3, the benchmarking and validation of simulations involving line-parallel methods and coarse-grain parallelism was presented. Additional improvements such as variable parallelism coupled with the use of data structures designed to keep the number of VRAM transactions at a minimum were also used in these simulations. In both 2D and 3D test-cases, reasonably high performance gains ( 30X) were demonstrated even when implicit systems were part of the solution procedure.

### 5.2.2 Hybrid Methodology

1. The central idea behind the hybrid method is the decomposition of the flow domain into two components and the use of different computational models inside these sub-domains. The choice of computational model in each sub-domain depends entirely on the nature of flow physics expected to be encountered there. In the case of brownout environments such as rotors hovering in ground effect, the demarcation of the sub-domains is fairly obvious. The region near the ground plane is expected to exhibit viscous

phenomena such as boundary layer growth and separation. In regions far removed from the ground plane, the flow can be assumed to be largely inviscid with vorticity confined to well-defined structures such as tip-vortices. In the present work, therefore, a hybrid methodology was developed that employed the numerical efficienct free-vortex method for regions far from the viscous ground plane and a relatively high fidelity RANS solver for the region near the ground.

2. The hybrid methodology developed in this work is meant to be an alternative to other modeling strategies such as overset meshes and the use of explicit source terms in the mesh-based solver. The disadvantage with this approach is that a first-principles approach is abandoned for the sake of numerical efficiency. Empirical constants such as initial vortex core sizes, vortex growth rates, coefficients of analytical impinging jet models, etc., need to be prescribed in this approach. Furthermore, the formulation of the hybrid method requires a well-defined separation between the two subdomains. For instance the use of the hybrid methodology to study the flow environment beneath a rotor operating in extreme ground effect, with the rotor height above the ground plane being of the same order as the viscous boundary layer height, would lead to difficulties. However, when properly chosen, the decrease in computational resources more than make up for the disadvantages.

3. For dual-phase simulations, the FVM-RANS hybrid was combined with a sediment tracking algorithm to study dispersed-phase characteristics. Both the free-wake solver and the sediment tracking algorithm are sub-classes of

the N-body problem and are handled using a fine-grain parallelism. Other extensions of the hybrid methodology are similarly possible, such as the use of a high-fidelity RANS solver to compute the airloads at the blades for a hovering rotor simulation.

### 5.2.3 Impinging Vortex Ring

1. The first validation test-case for the hybrid methodology was the experimental setup used by [81] and [17] to study a vortex ring impinging upon a ground plane. A free-vortex model is used to model the vortex ring while an analytical field is used to model the jet.

2. The comparison of predicted time-averaged radial velocity profiles with experimental data showed good agreement at all radial locations where the data was available. In particular, the peak wall-jet velocities at these radial locations were found to be well predicted. The agreement between experiment and prediction was found to deteriorate with increased distance from the ground plane.

3. The simulation also revealed the formation of a large, coherent secondary vortex at the ground plane. This vortex separation was observed to occur intermittently around $r/R = 2.0$. The strength of this vortex was measured to be approximately equal to the strength of the primary vortex that played a key role in its ejection. Similar observations of secondary vortices at the ground plane were reported in experimental studies by Geiser [81]. The size, position and relative strength of this secondary vortex was also found to be in good agreement with experiment.

### 5.2.4 Hovering Micro-scale Rotor

1. After gaining sufficient confidence in the capability of the hybrid solver to model single-phase flows, the methodology is employed to model the single and dual-phase flow environment around a hovering micro-scale rotor based on experiments by Sydney et al [16]. A free-vortex model is used to model the tip vortices. The strengths of these vortex filaments are prescribed based on airloads predicted by a linear aerodynamics model employing CFD-generated 2D airfoil tables. These airfoil tables are validated by comparing the predicted variation of sectional thrust with the predictions from a full-RANS simulation. Furthermore, the integrated thrust coefficient for this case were found to be within 5% of experimental measurements.

2. The comparison of predicted time-averaged radial velocity profiles with experimental data showed reasonable agreement at all radial locations where the data was available. At outer radial locations, the predicted values of peak jet velocity were observed to be higher than experimental measurements. This overprediction is due to the simulation producing vortices that were stronger than those observed in experiments which induce stronger velocities at the ground, thereby increasing the time-averaged values.

3. Predicted phase-averaged velocity profiles were also found to be in good agreement with experiments. The agreement between predictions and experimental observation deteriorated at outer radial locations, with the hybrid solver predicting much larger excursions in velocity in these regions. This can again be attributed to the stronger predicted vortex strength which in turn leads to more pronounced vortex signatures. In addition, there

seems to be a slight vertical offset in the position of the passing vortices at these radial locations.

4. Predicted friction velocities at the ground plane show that the maximum magnitudes of shear are observed at radial locations between 1.5R and 2.5R. This region coincides with the zone where the tip-vortices come into close proximity with the ground boundary layer. The local induced velocity of the tip vortex serves to energize the boundary layer leading to an increase in shear stresses. However, the resulting adverse pressure gradient at the wall was seen to result in flow separation and the creation of secondary vortices.

5. Next, the hybrid solver was coupled to a sediment tracking algorithm to study the dual-phase flow beneath a hovering rotor. The brownout simulation showed that among the heaviest particles (100 microns), the mechanism of creep was the most dominant with particles mostly creeping along the ground plane without being uplifted into suspension. This behaviour was also observed in experiments. One interpretation of this observation is that the larger inertia of these particles makes each particle a low-pass filter that reacts only to the time-averaged (lowest frequency) flow features which push them radially outward. The lightest particle set (1 microns) experienced no motion whatsoever; an observation similar to that reported in Ref. 16 where the lightest particles resulted in the least mobile bed due to large cohesive forces between particles of this scale. The intermediate sized particle set (50 micron) was found to be the most mobile with particles exhibiting many of the transport mechanisms observed in laboratory exper-

iments. In particular, the mechanisms of creep and vortex trapping were observed to occur during the simulation, with the latter mechanism being the dominant mode of transport at this scale. Furthermore, the region of maximum bed mobility was seen to coincide with the radial zones where the predicted friction velocity was observed to be the largest. In addition, the creation of sustained sediment waves outboard of the rotor was seen to occur in simulations - similar to those observed in [16].

6. The single-phase flow was seen to exhibit very low turbulence levels at these low Reynolds numbers. Switching off the turbulence model led to virtually identical velocity profiles near the ground plane.

7. The use of the roughness correction in the Spalart Allmaras turbulence model led to predictions of slightly higher turbulent stresses at the ground plane and consequently a more mobile sediment bed for both heavy and intermediate sized particles.

8. Due to memory constraints on modern GPU platforms grid convergence studies were difficult to perform. A 2X increase in the number of grid points in all directions would have rendered the simulation infeasible at the current technology level. So instead, two grids of different resolutions were tested - a baseline grid consisting of 60x180x120 points (a total of 1.7 million points with ghost cells included) and a fine-grid consisting of 120x180x120 points (a total of 3.3 million points with ghost cells included). In both cases, the velocity profiles were found to be virtually identical at all radial locations. In addition, three different wake discretizations were employed in the free-vortex domain: $5^o$, $2.5^o$ and $1.5^o$. In all three cases,

the induced velocities on the boundaries of the RANS mesh were seen to be indistinguishable suggesting that the wake discretization used in the present work is sufficiently fine.

## 5.3 Future Work

Some suggested future studies and potential improvements to the current methodology are listed here:

1. At present, the GPU-RANS solver makes use of only two levels of device memory - global memory and thread memory. The use of shared memory for certain bandwidth-intensive operations such as high-order, spatial reconstruction might help increase performance gains. Similarly, the use of constant, texture and pinned memory might result in better utilization of GPU resources and consequently, superior performance.

2. An analysis of the profile data from GPU-RANS simulations suggests that the bottleneck lies in the solution of implicit systems. Improving performance of these routines or investigating new candidate algorithms may help improve performance. One suggestion is the replacement of the Thomas Algorithm for tridiagonal inversion with a parallel inverter such as Parallel Cyclic Reduction (PCR).

3. At the current technology level, single GPU computations are memory-limited, rendering simulations involving large grids ($> 5$ million points) infeasible. One solution to this predicament is the extension of the hybrid

solver to run on multiple GPUs. If the methodology scales well across multiple cards, large, high resolution grids may be used to study high Reynolds number flow environments such as those encountered in full-scale rotorcraft brownout. One potential issue with multi-GPU computing is the latency involved in transferring interface information between GPUs. If the device-host-device route is employed for all inter-GPU transfers, large performance degradation may be possible. In contrast newer GPUs allow for a direct communication path between devices which has shown to result in less latency, but this stage of the process could still prove to be an expensive bottleneck.

4. Isolated rotor simulations yield valuable insight into the physics of brownout. However, to fully model rotorcraft brownout, the effect of the fuselage on both single and dual-phase flowfields needs to be considered. For modeling the fuselage either an immersed boundary method (IBC), an overset, body-fitted mesh or a source/vortex panel method may be used. A GPU-based IBC approach offers the most promise at the present time due to memory limitations on GPUs.

5. At higher Reynolds numbers, the effects of transition and turbulence become more significant. A sub-class of transition models are formulated as transport equations that can be implemented on the GPU using a coarse-grain mode of parallelism similar to that employed in the solution of the Spalart Allmaras equation.

6. All dual phase simulations in this study were conducted by assuming one-way coupling between the dispersed phase and the carrier phase. While

this assumption is defensible in the case of dilute flows, in regions near the boundary layer where particle concentrations can be relatively high, this simplification warrants revisiting. One possible strategy would be to add source-terms with magnitudes proportional to local particle concentrations. The effect of particle concentration on turbulent quantities will also need to be considered.

7. In the present study, the particle transport model was simplified by disabling the bombardment transport mechanism. In previous studies, this mechanism was found to have a significant effect on particle entrainment - a few high-momentum particles impinging upon the ground plane were capable of mobilizing a much larger number of particles. The difficulty with implementing the bombardment mechanism on GPU platforms lies in the process by which collisions (between particles that are mobilized and particles that are stationary) are identified. This requires the use of data-dependent conditionals with the potential for significant warp-divergence. Any future study that seeks to include this mechanism would require an efficient and GPU-amenable, spatial partitioning scheme with fewer conditionals and highly coalesced data transfers between VRAM and chip.

# Appendices

# Appendix A

# Key CUDA Abstractions

As mentioned in chapter 1, three key abstractions lie at the heart of CUDA.

1. A hierarchy of thread groups

2. Shared memory

3. Barrier synchronization

## A.1 Thread Hierarchy

The first abstraction represents the idea that a problem can be divided up into smaller 'threads' , each of which can be handled by a separate arithmetic unit. As an example of this abstraction, consider the simple arithmetic operation SAXPY (Single-precision real Alpha X Plus Y) . SAXPY is a combination of scalar multiplication and vector addition,

$$\mathbf{y} \leftarrow \alpha\mathbf{x} + \mathbf{y} \tag{A.1}$$

where $\alpha$ is a scalar, and $\mathbf{x}$ and $\mathbf{y}$ are vectors.

Shown below is a CPU implementation of this operation, written in C.

```
// SAXPY.c
void saxpy(float* x, float* y, int n, float a) {
  int i;
  for (i = 0; i < n; ++i)
    {
    y[i] += a * x[i];
    }
}
```

In contrast, consider the GPU implementation of SAXPY, written in CUDA.

```
// SAXPY.cu
__global__ void saxpy(float* x, float* y, int n, float a) {
  int i;
  i = blockDim.x * blockIdx.x + threadIdx.x;
  while (i<n)
  {
    y[i] += a * x[i];
    i += blockDim.x*gridDim.x;
  }
}
```

The first difference between the two code snippets is the presence of the declaration specifier 'global' in the CUDA code before the function signature. This qualifier implies that the function (called a 'kernel') will be executed on the GPU device. The second key difference is that in the CPU version, the counter $i$ sequentially from 0 to the array limit inside a for-loop. In contrast, the GPU version initializes the variable i to the thread index given by $blockDim.x * blockIdx.x + threadIdx.x$ where $blockDim.x$ refers to the number of threads per block and $blockIdx.x$ refers to the 0-indexed ID of the current block and $threadIdx.x$ refers to the 0-indexed ID of the current thread within this block. When this function is called, a 'grid' of blocks, with each block having a set of threads is spawned to perform the required operation. These grids and blocks can be 1-dimensional, 2-dimensional or 3-dimensional. In the example shown above, a 1D grid of blocks and 1D block of threads is used for clarity. To call the above function, the number of threads per block and the number of blocks per grid needs to be specified. The following function call is an example:

```
saxpy<<<10,10>>>(x, y, int 100, 5.0)
```

The execution configuration <<<>>> is used to specify the number of blocks per grid and the number of threads per block. In the above example, to perform a SAXPY operation on an array with 100 elements, a grid of 10 blocks is used, with each block consisting of 10 threads.

Each thread is aware of its own block index and thread index within the block. The for-loop which was present in the CPU version of the code is replaced with

a while loop in the CUDA version with each thread checking to see if its own ID lies between array limits. If this conditional returns a positive, the thread then operates upon the array entry that has an index matching the thread index. In the event that the total number of threads spawned is smaller than the number of array elements to be operated upon, a single thread is mapped to multiple array indices that are separated from each other by a stride length equal to $blockDim.x * gridDim.x$

## A.2 Shared Memory

CUDA threads are capable of accessing data from multiple memory locations during their execution as shown in Figure A.1. Each thread has its own private local memory which has the same lifetime as the thread. All threads belonging to the same block have access to shared memory which has the same lifetime as the block. All blocks of threads have access to the same global memory which also happens to be the biggest unit of memory on the device.

In addition to the above, there are also two additional read-only memory spaces available to all threads: constant and texture memory. Each level of the memory hierarchy has its own latency when data retrieval is performed. This latency is typically inversely proportional to the size of memory available. For instance, the amount of global memory in modern GPUs can be as high as several GB. But data retrieval has a latency of about 400-800 clock cycles. For this reason, to achieve significant performance gains over serial computations, the number of global memory transactions needs to be kept at a minimum.

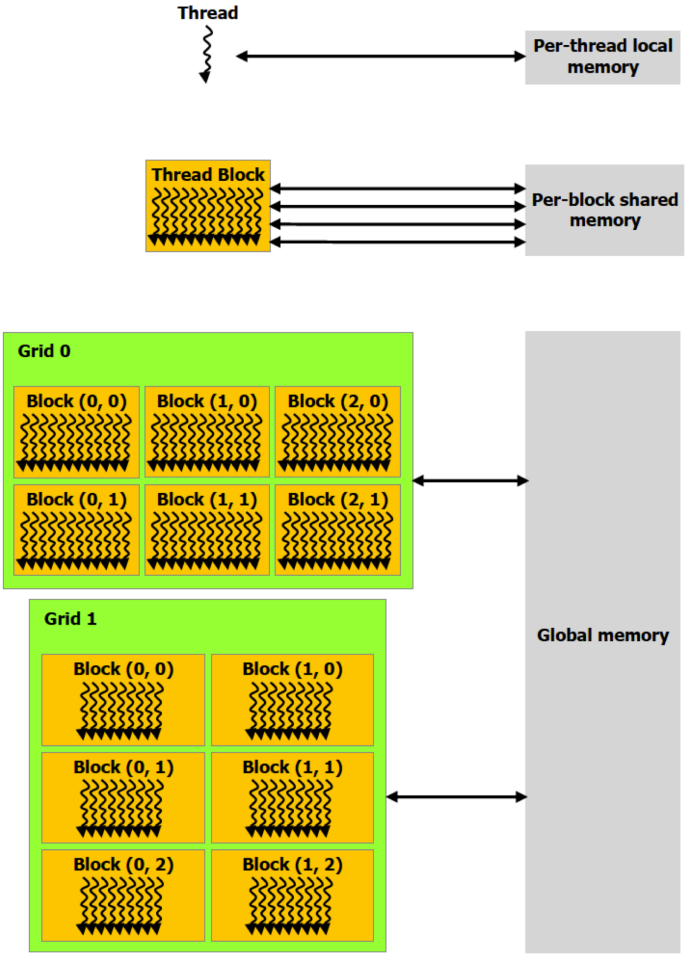



**Figure A.1:** Memory hierarcy available on modern GPUs

## A.3 Barrier Synchronization

One important abstraction that CUDA allows is the idea of barrier synchronization to control the execution of threads. Certain classes of vector operations require at each step, that all threads have completed their individual tasks before progressing. One example of such a class is the 'reduce' operation where an array of elements are combined using an associative operator like addition, XOR or multiplication. At each step of the reduce operation, it is important that all the array elements to be operated upon have already been updated in the previous step. Hence it is important that all the threads that participated in step $i$ of the reduce operation have completed execution before step $i + 1$ begins. There are two ways this can be achieved:

1. Packaging functionality to ensure that successive steps that require thread synchronization between them are mapped to separate kernel invocations. Each time, a kernel terminates, all participating threads are automatically synchronized.

2. Using the *syncthreads*() intrinsic function if all the threads within a single block need to be synchronized

# Appendix B

# Recurrence Relations for Associated Laguerre Polynomials

In Section 2.7.3, an analytical field [67] for an impinging jet was described. This field uses a series consisting of Laguerre polynomials to approximate the flowfield. The computation of Laguerre polynomials can be computationally expensive. Fortunately, there exists a set of recurrence relations for the Laguerre polynomials and their derivatives that can be used to compute the series in a numerically efficient way.

The constant $c_n$ in Eqn. 2.61 can be computed using:

$$c_{n+1} = -c_n \frac{n}{(n+1)^2} \tag{B.1}$$

The associated Laguerre polynomials can be computed using the following

recurrence formula:

$$L_n^1 = \frac{n}{n-1}[(2(n-1) - \frac{2r^2}{k})L_{n-1}^1 - (n-1)^2 L_{n-2}^1] \tag{B.2}$$

The above relation is applicable for $n > 2$ with initial terms,

$$\begin{aligned} L_1^1 &= -1 \\ L_2^1 &= 2(\frac{2r^2}{k} - 2) \end{aligned} \tag{B.3}$$

The derivatives of the Laguerre polynomial can be computed using:

$$\begin{aligned} \frac{\partial L_n^1}{\partial r} &= \frac{n}{n-1}[(2(n-1) - \frac{2r^2}{k})\frac{\partial L_{n-1}^1}{\partial r} - (n-1)^2\frac{\partial L_{n-2}^1}{\partial r}] \\ &- \frac{4nr}{(n-1)k}L_{n-1}^1 \end{aligned} \tag{B.4}$$

The above relation is applicable for $n > 2$ with initial terms,

$$\begin{aligned} \frac{\partial L_1^1}{\partial r} &= 0 \\ \frac{\partial L_2^1}{\partial r} &= \frac{8r}{k} \end{aligned} \tag{B.5}$$

# References


[1] Syal, M., *Development of a Lagrangian-Lagrangian Methodology to Predict Brownout Dust Clouds* PhD thesis, University of Maryland, 2012.

[2] Mapes, P., Kent, R., and Wood, R., "DOD Helicopter Mishaps FY85-05: Findings and Recommendations," US Air Force, 2008.

[3] Jansen, C., Wennemers, A., and Groen, E., "FlyTact: A Tactile Display Improves a Helicopter Pilots Landing Performance in Degraded Visual Environments," *Haptics: Perception, Devices and Scenarios*, Vol. 502, No. 4, 2008, pp. 867–875.

[4] Cowherd, C., "Sandblaster 2: Support of See-Through Technologies for Particulate Brownout," US Army Aviation and Missile Command, 2007.

[5] Pickford, M., "Operating Helicopters Safely in a Degraded Visual Environment in Support of Military Operations," Proceedings of the Royal Aeronautical Society, Rotorcraft Group Conference, London, UK, June 16–17 2010, Royal Aeronautical Society, Rotorcraft Group.

[6] Milluzzo, J., and Leishman, J. G., "Assessment of Rotorcraft Brownout

Severity in Terms of Rotor Design Parameters," *Journal of the American Helicopter Society*, 2010.

[7] "The Global Elite Forces and Special Operations Webpage,", 2011.

[8] Wong, O., and Tanner, P., "Photogrammetric Measurements of an EH-60L Brownout Cloud," 66th Annual Forum Proceedings of the American Helicopter Society, Pheonix, AZ, May 10–13 2010.

[9] Taylor, M. K., "A Balsa-Dust Technique for Air-Flow Visualization and its Application to Flow through Model Helicopter Rotors in Static Thrust," NACA NACA TN-2220, Nov. 1950.

[10] Curtiss, H. C., Sun, M., Putman, W., and Hanker, E. J., "Rotor Aerodynamics in Ground Effect at Low Advance Ratios," *Journal of the American Helicopter Society*, Vol. 29, No. 1, 1984, pp. 48–55.

[11] Light, J., "Tip Vortex Geometry of a Hovering Helicopter Rotor in Ground Effect," *Journal of the American Helicopter Society*, Vol. 38, No. 2, 1993, pp. 34–42.

[12] Nathan, N., and Green, R., "Measurements of a Rotor Flow in Ground Effect and Visualisation of the Brownout Phenomenon," 64th Annual Forum Proceedings of the American Helicopter Society, Montréal Canada, April 29–May 1 2008.

[13] Lee, T., Leishman, J. G., and Ramasamy, M., "Fluid Dynamics of Interacting Blade Tip Vortices with a Ground Plane," *Journal of the American Helicopter Society*, Vol. 55, 2010.

[14] Johnson, B., Leishman, J. G., and Sydney, A., "Investigation of Sediment Entrainment using Dual-Phase, High-Speed Particle Image Velocimetry," *Journal of the American Helicopter Society*, Vol. 55, 2010.

[15] Milluzzo, J., Sydney, A., Rauleder, J., and Leishman, J. G., "In-Ground-Effect Aerodynamics of Rotors with Different Blade Tips," 66th Annual Forum Proceedings of the American Helicopter Society, Phoenix, AZ, May 10–13 2010.

[16] Sydney, A., Baharani, A., and Leishman, J. G., "Understanding Brownout Using Near-Wall Dual-Phase Flow Measurements," 67th Annual Forum Proceedings of the American Helicopter Society, May 3–5 2011.

[17] Mulinti, R., and Kiger, K., "Two-phase PIV Measurements of Particle Suspension in a Forced Impinging Jet," Proceedings of the 63rd Annual Meeting of the APS Division of Fluid Dynamics, November 2010, Vol. 55.

[18] Dade, B., "Near-Bed Aeolian Sediment Transport Under Non-Uniform Flows," Proceedings of the American Geophysical Union, December 2009.

[19] Syal, M., "Modeling of Bombardment Ejections in the Rotorcraft Brownout Problem," *Journal of Aircraft*, Vol. 51, No. 4, 2013.

[20] Syal, M., Govindarajan, B., and Leishman, J. G., "Mesoscale Sediment Tracking Methodology to Analyze Brownout Cloud Developments," 66th Annual Forum Proceedings of the American Helicopter Society, Phoenix, AZ, May 11–14 2010.

[21] Wachspress, D., Whitehouse, G., Keller, J., Yu, K., Gilmore, P., Dorsett, M., and McClure, K., "High Fidelity Rotor Aerodynamic Module for Real Time Rotorcraft Flight Simulation," Proceedings of the 65th Annual American Helicopter Society Forum, Grapevine, TX, May 27–29 2009.

[22] Phillips, C., Kim, H., and Brown, R. E., "The Flow Physics of Helicopter Brownout," Proceedings of the 66th Annual American Helicopter Society Forum, Pheonix, AZ, May 11–14 2010.

[23] Wenren, Y., Walter, J., Fan, M., and Steinhoff, J., "Vorticity Confinement and Advanced Rendering to Compute and Visualize Complex Flows," 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan. 9–12 2006.

[24] Kalra, T. S., Lakshminarayan, V. K., and Baeder, J. D., "CFD Validation of Micro Hovering rotor in Ground Effect," 66th Annual Forum Proceedings of the American Helicopter Society, Pheonix, AZ, May 11–14 2010.

[25] Morales, F., "Euler-Lagrange Modeling of Vortex Interaction with a Particle-Laden Turbulent Boundary Layer," Master's thesis, Arizona State University, May 2011.

[26] Thomas, S., Lakshminarayan, V. K., Kalra, T. S., and Baeder, J. D., "Eulerian-Lagrangian Analysis of Cloud Evolution using CFD Coupled with a Sediment Tracking Algorithm," Proceedings of the 67th Annual Forum of the American Helicopter Society International, Virginia Beach, VA, May 2–5 2011.

[27] "The MPI Forum: The MPI Message-passing Interface Standard *http://www.mcs.anl.gov/mpi/standard.html*,", May 1995.

[28] Chapman, B., Jost, G., and van der Pas, R., *Using OpenMP* MIT Press, 2007.

[29] *NVIDIA CUDA C Programming Guide* NVIDIA, 2012.

[30] "The Top 500 Supercomputer Sites *www.top500.org*,", June 2013.

[31] Gumerov, N. A., and Duraiswami, R., "Fast Multipole Methods on Graphics Processors," *Journal of Computational Physics*, 2008, pp. 8290–8313.

[32] Hu, Q., Gumerov, N. A., Duraiswami, R., Syal, M., and Leishman, J. G., "Toward Improved Aeromechanics Simulations Using Recent Advancements in Scientific Computing," Proceedings 67th Annual Forum of the American Helicopter Society, Virginia Beach, VA, May 3–5 2011.

[33] Merrill, D., Garland, M., and Grimshaw, A., "High Performance and Scalable GPU Graph Traversal," University of Virginia Technical Report CS-2011-05, VA, August 2011.

[34] Harish, P., and Narayanan, P. J., "Accelerating Large Graph Algorithms on the GPU using CUDA," Proceedings of the IEEE High Performance Computing Conference, Goa, India, December 18 – 21 2007.

[35] Vineet, V., Harish, P., Patidar, S., and Narayanan, P. J., "Fast Minimum Spanning Tree for Large Graphs on the GPU," Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on High Performance Graphics, New Orleans, LA, August 1–3 2009.

[36] Hagen, T. R., Lie, K. A., and Natvig, J. R., "Solving the Euler Equations on

Graphics Processing Units," *International Conference for Computer Science*, Vol. 3994, 2006, pp. 220 – 227.

[37] Kestener, P., Chateau, F., and Teyssier, R., "Accelerating Euler Equations Numerical Solver on Graphics Processing Units," Proceedings of the International Workshop on High Performance Computing Technologies and Applications, Busan, Korea, May 21–23 2010.

[38] Stock, M. J., and Gharakhani, A., "A GPU-accelerated Boundary Element Method and Vortex Particle Method," Proceedings of the 40th AIAA Fluid Dynamics Conference and Exhibit, Chicago, IL, June 28 – July 1 2010.

[39] Thibault, J., and Senocak, I., "CUDA Implementation of a Navier-Stokes on Multi-GPU Desktop Platforms for Incompressible Flows," Proceedings of the 47th AIAA Aerospace Sciences Meeting, Orlando, FL, January 2009.

[40] Chandar, D. D. J., Sitaraman, J., and Mavriplis, D., "CU++ET: An Object Oriented Tool for Accelerating Computational Fluid Dynamics Codes Using Graphical Processing Units," Proceedings of the 20th AIAA Computational Fluid Dynamics Conference, Honolulu, HI, June 27–30 2011.

[41] Chandar, D. D. J., Sitaraman, J., and Mavriplis, D., "GPU Parallelization of an Unstructured Overset Grid Incompressible Navier-Stokes Solver for Moving Bodies," Proceedings of the 50th AIAA Aerospace Sciences Meeting, Nashville, TN, January 9–12 2012.

[42] Stone, C. P., Duque, P. N., Zheng, Y., Car, D., Owens, J. D., and Davis, R. L., "GPGPU Parallel Algorithms for Structured-Grid CFD Codes," Pro-

ceedings of the 20th AIAA Computational Fluid Dynamics Conference, Honolulu, HI, June 27–30 2011.

[43] Jesperson, D. C., "Acceleration of a CFD code with a GPU," *Scientific Programming - Exploring Languages for Expressing Medium to Massive On-Chip Parallelism*, Vol. 18, August 2010 2010, pp. 193–201.

[44] Zechner, M., and Granitzer, M., "Accelerating K-Means on the Graphics Processor via CUDA," Proceedings of the First International Conference on Intensive Applications, 2009.

[45] Corrigan, A., Camelli, F., Wallin, J., and Lohner, R., "Running Unstructured Grid Based CFD Solvers on Modern Graphics Hardware," Proceedings of the 19th AIAA Conference on Computational Fluid Dynamics, San Antonio, June 2009.

[46] Zhang, S. S., Heck, M., and Barthelemy, P., "Multi GPU Accelerated Wing Tip Vortex Extraction," Proceedings of the 20th AIAA Computational Fluid Dynamics Conference, Honolulu, HI, June 27–30 2011.

[47] O'Neil, M. A., Tamir, D., and Burtscher, M., "A Parallel GPU Version of the Traveling Salesman Problem," International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV, July 18–21 2011.

[48] Gingold, R., and Monaghan, J., "Smoothed particle hydrodynamics: theory and application to non-spherical stars," *Journal of the Royal Astronomical Society*, Vol. 181, 1977, pp. 375–389.

[49] Bhagwat, M. J., and Leishman, J. G., "Time-Accurate Modeling of Rotor Wakes Using A Free-Vortex Wake Method," AIAA CP 2000-4120, 18$^{\text{th}}$ AIAA Applied Aerodynamics Conference, Denver, CO, August 2000.

[50] McDonald, P., "The Computation of Transonic Flow through Two-Dimensional Gas Turbine Cascades," Proceedings of the Annual Conference of American Society of Mechanical Engineers, 1971.

[51] B., V. L., "Towards the Ultimate Conservative Difference Scheme V: A Second Order Sequel to Godunovs Method," *Journal of Computational Physics*, Vol. 32, No. 101, 1979.

[52] Jiang, G., and Shu, C., "Efficient Implementation of Weighted ENO Schemes," *Journal of Computational Physics*, Vol. 126, No. 202, 1996.

[53] Ghosh, D., *Compact-Reconstruction Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Conservation Laws* PhD thesis, University of Maryland, 2012.

[54] Roe, P., "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes," *Journal Computational Physics*, Vol. 135, No. 2, 1997, pp. 250–258.

[55] Baldwin, B., and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Flows," Proceedings of the $16^{th}$ AIAA Aerospace Sciences Meeting, Huntsville, AL, January 1978.

[56] Durbin, P., "A Reynolds Stress Model for Near Wall Turbulence," *Journal of Fluid Mechanics*, Vol. 249, 1993, pp. 465–498.

[57] Srinivasan, G., Ekaterinaris, J., and McCroskey, W. J., "Evaluation of Turbulence Models for Unsteady Flows of an Oscillating Airfoil," *Computers and Fluids*, Vol. 24, No. 7, 1995, pp. 833–861.

[58] Spalart, P. R., and Almaras, S. R., "A One Equation Turbulence Model for Aerodynamics Flow," American Institute of Aeronautics and Astronautics 1992-0439, 1992.

[59] Aupoix, B., and Spalart, P. R., "Extensions of the Spalart-Allmaras Turbulence Model to Account for Wall Roughness," *International Journal of Heat and Fluid Flow*, Vol. 24, 2003.

[60] Hirsch, C., *Numerical Computation of Internal and External Flows, Volume 2* Wiley Publishers, 1990.

[61] Pulliam, T., H., and Chaussee, D., "A Diagonal Form of an Implicit Approximate Factorization Algorithm," *Journal of Computational Physics*, Vol. 39, 1981.

[62] Pulliam, T., H., *Solution Methods in Computational Fluid Dynamics* NASA Ames Research Center.

[63] Sitaraman, J., and Roget, B., "Prediction of Helicopter Maneuver Loads Using A Coupled CFD/CSD Analysis," $26^{\text{th}}$ International Congress of the Aeronautical Sciences, 2008.

[64] Sitaraman, J., Baeder, J., and Iyengar, V., "On the Field Velocity Approach and Geometric Conservation Law for Unsteady Flow Simulations," AIAA

Paper 2003-3835, 16th AIAA Computational Fluid Dynamics Conference, 2003.

[65] Gopalan, G., Sitaraman, J., Baeder, J., and Schmitz, F., "Assessment of Aerodynamic and Aeroacoustic Prediction Methodologies with Application to the HART II Model Rotor," 62nd American Helicopter Society International Forum and Technology Display, Phoenix, AZ, May 2006.

[66] Vatistas, G. H., Kozel, V., and Mih, W. C., "A Simpler Model for Concentrated Vortices," *Experiments in Fluids*, Vol. 11, 1991, pp. 73–76.

[67] Xu, Z., Hangan, H., and Yu, P., "Analytical Solutions for a Family of Gaussian Impinging Jets," *Journal of Applied Mechanics*, Vol. 75, 2008.

[68] Bagnold, R., *The Physics of Blown Sand and Desert Dunes* Dover Publications Inc., 1941.

[69] Amiraux, M., Thomas, S., and Baeder, J. D., "Fuselage Modelization for Multi-Fidelity Coupled CFD/CSD Simulation of Rotorcrafts in BVI Condition," Proceedings of the $31^{st}$ AIAA Applied Aerodynamics Conference, San Diego, CA, June 2013.

[70] Inoue, O., and Hattori, Y., "Sound generation by shock vortex interactions," *Journal of Fluid Mechanics*, Vol. 380, No. 81, 1999.

[71] Zhang, S., Zhang, Y., and Shu, C., "Multistage interaction of a shock wave and a strong vortex," *Physics of Fluids*, Vol. 17, No. 116101, 2005.

[72] Chatterjee, S., and Vijayaraj, S., "Multiple sound generation in interaction

of shock wave with strong vortex," *Journal of Aircraft*, Vol. 46, No. 2558, 2008.

[73] Cook, P., McDonald, M., and Firmin, M., "Aerofoil RAE 2822 - Pressure distributions, and boundary layer and wake measurements (Experimental Data Base for Computer Program Assessment, AGARD Report AR 138, 1979).," North Atlantic Treaty Organization Advisory Group for Aerospace Research and Development, 1979.

[74] Tatsumi, S., Martinelli, L., and Jameson, A., "A new high resolution scheme for compressible flows past airfoil," *Journal of Aircraft*, No. 95-0466, 1995.

[75] Schmitt, V., and Charpin, F., "Pressure distributions on the ONERA-M6-wing at transonic Mach numbers (Experimental Data Base for Computer Program Assessment, AGARD AR 138, 1979).," North Atlantic Treaty Organization Advisory Group for Aerospace Research and Development, 1979.

[76] Freeman, C., and Mineck, R., "Fuselage Surface Pressure Measurments of a Helicopter Wind-Tunnel Model with a 3.15-meter Diameter Single Rotor," Langley Research Center, 1979.

[77] Phelps, A., and Berry, J., "Description of the U. S. Army Small-Scale 2-Meter Rotor Test System," Langley Research Center, 1987.

[78] Mineck, R., and Gorton, S., "Steady and Periodic Pressure Measurements on a Generic Helicopter Fuselage Model in the Presence of a Rotor," Langley Research Center, 2000.

[79] Renaud, T., O'Brien, D., Smith, M., and Potsdam, M., "Evaluation of Isolated Fuselage and Rotor-Fuselage Interaction Using CFD," Proceedings 60th Annual Forum of the American Helicopter Society, June 2004.

[80] Shaeffler, N., Allan, B., Lienard, C., and Le Pape, A., "Progress Towards Fuselage Drag Reduction via Active Flow Control: A Combined CFD and Experimental Effort," Proceedings of the $36^{th}$ European Rotorcraft Forum, May 2010.

[81] Geiser, J., *Effects Of Wall Plane Topology on Vortex-Wall Interactions in a Forced Impinging Jet* PhD thesis, University of Maryland, 2012.

[82] Lakshminarayan, V. K., *Computational Investigation of Micro-Scale Coaxial Rotor Aerodynamics in Hover* PhD thesis, University of Maryland, 2009.

[83] Camenen, B., Bayram, A., and Larson, M., "Equivalent Roughness Height for Plane Bed under Steady Flow," *Journal of Hydraulic Engineering*, Vol. 132, No. 11, 2006.

[84] Thomas, S., Kalra, T. S., and Baeder, J. D., "A Hybrid CFD Methodology to Model the Two-phase Flowfield beneath a Hovering Laboratory Scale Rotor," Proceedings of the 42nd AIAA Fluid Dynamics Conference, New Orleans, LA, June 2012.